DOCUMENT RESUME

ED 357 732 IR 016 090

TITLE	Association of Small Computer Users in Education (ASCUE) Summer Conference. Proceedings (25th, North Myrtle Beach, South Carolina, June 21-25, 1992).
INSTITUTION	Association of Small Computer Users in Education, Greencastle, IN.
PUB DATE	Jun 92
NOTE	306p.
PUB TYPE	Collected Works - Conference Proceedings (021) -- Reports - Descriptive (141) -- Reports - Research/Technical (143)
EDRS PRICE	MF01/PC13 Plus Postage.
DESCRIPTORS	Computer Assisted Instruction; Computer Literacy; Computer Networks; Computer Simulation; Computer Software Development; *Computer Uses in Education; Conference Papers; *Curriculum Development; Delivery Systems; Elementary Secondary Education; Higher Education; Hypermedia; *Information Technology; *Microcomputers; *Telecommunications
IDENTIFIERS	Computer Mediated Communication; Internet

ABSTRACT

Forty-three papers from a conference on microcomputers are presented under the following headings: Computing in the Curriculum, Information and Computer Science Information; Institutional and Administrative Computing, and Management, Services, and Training. Topics of the papers include the following: telecommunications projects that work in university classrooms of tomorrow; providing voice, data, and video communications to local schools; computers in calculus and throughout the mathematics curriculum; computer modeling in traditional classrooms; multimedia in education; cross-curricular computing in a liberal arts college; using hypermedia to develop a political science simulation; using an integrated instructional delivery system for remediation in higher education; computer-mediated communication in curricula; teaching elementary matrix operations using spreadsheets; case studies in systems analysis; teaching data communications; teaching programming with closed laboratories; the evolution of information systems organization; the evolution of a software engineering course; the changing role of computing in college administration; learning to collaborate; a computerized academic advising package; developing a LAN-based student information system; human resources information systems; a workshop for faculty about Internet services; promoting computer literacy among faculty and academic staff; faculty and staff personal computer training; effective use of student employees in a microcomputer laboratory; and multi-user access in a campus network. Many of the papers contain references. (SLD)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

IR

ED357732

ASCUE

25th Anniversary 1967-1992

ASSOCIATION OF SMALL COMPUTER USERS IN EDUCATION

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☐ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy

Proceedings of the 1992 ASCUE Summer Conference

25th Annual Conference
June 21-25, 1992

North Myrtle Beach, South Carolina

BEST COPY AVAILABLE

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

Carl Singer

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

ASCUE TUG

CUETUG (College and University Eleven Thirty Users Group) 1967-1975

112016090

ASSOCIATION OF SMALL COMPUTER USERS IN EDUCATION (ASCUE)

1991-1992 Board Members with Business Addresses and Telephone Numbers E-mail address (years remaining in office)

PRESIDENT
(1 yr.)
Bill Wilson
Gettysburg College
Academic Computer Center
Gettysburg, PA 17325
(717) 337-6933
wilson@gburg (bitnet)
wilson@gettysburg.edu (internet)

TREASURER
(1 yr.)
Thomas Pollack
Duquesne University
School of Business & Admin.
Pittsburgh, PA 15282
(412) 434-1639

IMMED. PAST PRESIDENT
(1 yr.)
Dave Redlawsk
Rutgers University
Box 2237
East Millstone, NJ 00875
(908) 281-0924
redlawsk@zodiak (bitnet)
redlawsk@zodiak.rutgers.edu

PUBLIC RELATIONS
(1 yr.)
R. Waldo (Wally) Roth
Computer & System Science Dept.
Taylor University
Upland, IN 46989
(317) 998-5269

PROGRAM CHAIRMAN
(1 yr.)
Carl Singer
Computing Services
DePauw University
Greencastle, IN 46135
(317) 658-4294
singer@depauw

SECRETARY
(2 yrs.)
Dagrun Bennett
Computing Services
Franklin College
Franklin, IN 46131
(317) 738-8150
bennettd@depauw

LIBRARIAN/HISTORIAN

Jack Cundiff
Horry-Georgetown Tech. Coll.
P.O Box 1966
Conway, SC 29526
(803) 448-8506 (call between 8-9 a.m. Eastern time)

BOARD MEMBER AT LARGE
(1 yr.)
Frederick (Rick) Huston
Univ. of S.Carolina-Aiken
171 University Park
Aiken, SC 29801
(803) 648-6851

BOARD MEMBER AT LARGE
(2 yrs.)
Mary V. Connolly
Mathematics Department
Saint Mary's College
Notre Dame, IN 46556
(219) 284-4497

Table of Contents

Computing in the Curriculum

Telecommunications Projects that Worked in the University Classroom of Tomorrow, 1991-92. Marvin Bratt 1

Meeting the CHALLENGE: Providing voice, data, and video communications to local primary and secondary schools. Alessandro C. Calianese 6

Computers In Calculus (and Throughout the Mathematics Curriculum). G. Daniel Callon 20

Computer Modeling in the Traditional Classroom. Peter L. Havholm, Larry L. Stewart 24

Multimedia in Education. Carl W. Helms, Doris R. Helms 34

Cross-Curricular Computing in a Liberal Arts College. James A Koger 38

"This Stuff's Not Contagious!" Why? And What Can We Do About It? Joseph McDonald, Lynda Micikas 42

Using Hypermedia to Develop a Political Science Simulation. David P. Redlawsk 50

Using Information Technology for the Remediation of Students in Higher Education - Integrated Instructional Delivery System -. Man M. Sharma 65

Computer-mediated communication in two curricula: Exploring a telecommunications literacy. Matthew Stuve, Joan Rose, Donna Satterfield 74

Teaching Elementary Matrix Operations Using Spreadsheets. David S. Tucker .. 82

Information and Computer Science Education

Case Studies in Systems Analysis and Design Course and Senior Project Course. Jill Gray 94

Teaching Data Communications. Louis M. Janelle, Jr. 98

Teaching Programming with Closed Laboratories. David E. Maharry 99

File Structures Using C. Patricia T. Montague, James R. Van Doren 104

A C.A.S.E. for Systems Implementation. Paul Moriarty 124

The Evolution of the Information Systems Organization. Thomas A. Pollack 128

Out from the Swamp: The Evolution of a Software Engineering Course. Robert L. Sedlmeyer 132

A Closed Laboratory in CS-1 Using Tutorials. Peter D. Smith 138

A Laboratory Component for CS1. Thomas J. Smith, Charles H. Stolze 148

Institutional and Administrative Computing

The Changing Role of Computing in College Administration. John A. Anderson 152

Learning to Collaborate. William Doemel 160

Computerized Academic Advising Package. Valerian S. M. John, Gurbax Singh 164

Safe Computing: Protection Against Viruses. Larry D. Jones 169

Developing a Lan-based Student Information System. Larry D. Jones, Mel Anthony May 173

Human Resource Information Systems: An ASCUE personnel profile. William J. Moressi 179

Real-time On-line Billing System For a University Environment. Balwant G. Singh, Paul G. Masucci 189

Computer Based Student Advising. Thomas J. Smith 196

Reengineering Campus Administrative Information Systems. Glen W. Turney ... 203

Earning "Monopoly Rents" Through Innovative Programs Using a PC. Mary Wall 214

Management, Services, and Training

A Workshop for Faculty about Internet Services. Ernest C. Ackermann 217

Educating the faculty and staff to computer use: A working model. Donald Armel, Robert Barger 227

Helping Adults Learn. Planning and Implementing Effective Inservice Programs at a Small Liberal Arts College. Elizabeth Lancaster Braswell 232

Developing and Integrating Computer Labs in Non Computer Courses. Francis X. Foley 238

Promoting Computer Literacy Among Faculty and Academic Staff: The Approach at a Small Liberal Arts College. Patrick Kopp, Joan Lescinski 241

A Practical Approach to Faculty and Staff Personal Computer Training. Mick Lantis 246

Developing User Training/Support: Hints for Success. Jeanne V. Livers 249

How to Support Computer Labs for 5000 students with a Staff of One. Kathryn McCubbin 252

Effective Use of Student Employees In A Microcomputer Lab. An Alternate Two-Class System. Michael J. Payne 257

Multi-user Access In a Campus Network: The Next Generation. Mary L. Smalls, Stephen E. Glover 264

Utilizing Computer Labs Across the Technology and Other Curriculums. Richard L. Taylor 279

Empowering Student Employees in a Down-Side Economy. Stephen A. Vieira . . . 289

Connecting to the Internet: A Look at the Services and the Tools to Use Them. William P. Wilson. 293

Telecommunications Projects that Worked in the University Classroom of Tomorrow, 1991-92

Dr. Marvin Bratt
College of Education
Ohio State University, Marion Campus
1465 Mt. Vernon Avenue
Marion, OH 43302
hbratt@magnus.acs.ohio-state.edu

Introduction

In this paper I will describe several projects that my students and I have participated in during the 1991-92 school year that involve telecommunications. I will describe two major projects and several smaller ones.

For the past three years I have directed the University Classroom of Tomorrow, a project funded by the Ohio Board of Regents under their Academic Challenge Program. This project is an experimental program for elementary education majors that lasts four years. The students take a three hour introductory course and then follow it up with one hour seminars each quarter until they graduate. One of the projects we work with is telecommunications. Currently, we subscribe to Learning Link, a service of WVIZ public television in Cleveland, Ohio. I am a monitor for the Macintosh and Apple user groups on Learning Link. We also subscribe to the Apple Global Education project that runs on AppleLink, corporate network of Apple Computer. I am part of the Research and Development team in the Advanced Technology Group at Apple. Each of the UCOT students and staff has an account on MAGNUS, Ohio State's email system that is connected to INTERNET. Currently, I monitor the Hypercard and KidsNet discussion lists and rea d the Distance Education Online Symposium as well.

Learning Link Projects

Our connection with Learning Link is primarily to schools in Ohio. There are currently over 900 classrooms in Ohio connected. Our purpose for this project was to get the students started with telecommunications using local schools that they also participate in. Three local schools are members of Learning Link and thus we can communicate directly with them using email and other services available. The UCOT project provides five portable computers for the students' use. Two of these have modems built in and can be used for telecommunications. Learning Link is relatively easy to use, is generic and works on most computers, allows for unlimited usage (with user fee) and utilizes an 800 phone number which means that our students and staff can call into it from home at any time.

One of the best projects we have discovered with Learning Link is access to Cable Network News' Newsroom program. Our students can tape record a specially prepared presentation of CNN that airs at 3:45 am, log onto Learning Link the next morning and find several sets of documents pertaining to the session prepared by "Teachable Tech." It is an excellent source of information for social studies education, current events, and politics. We also have used Learning Link to correspond with schools outside our district. One recent discussion concerned a debate over whether coaches should also be classroom teachers. Another area is the general discussion area that my students use frequently. My students are just beginning their training to be teachers. Many have had limited experiences with children, but few have had extended interactions such as can be done on a network. This provides a very good outlet for their questions and explorations without risk.

As monitor for the Macintosh forum, I answer questions regarding hardware and software that users have. Common questions include queries about new computers, what computers to purchase, what types of modem software is helpful, how to hook up local area networks, etc. We had a small group of teachers

who were exchanging news items for a time. I think with some student involvement, this project could take off. This project included schools in Indiana as well as Ohio.

INTERNET and MAGNUS

MAGNUS is our email system at Ohio State. Our students learn to use MAGNUS during their second year of study. We use it for all sorts of things from email to submission of projects. MAGNUS allows us to search our library system for the university as well as the State of Ohio library and approximately 35 other libraries that have sharing relationships with Ohio State. In addition to email, we can read the Clarion news service, follow the usenet newsgroup programs and communicate with other host computers using FTP (file transfer protocol). Students who discover "Fetch" can link to INTERNET's public access servers and download graphic image files, sound files, public domain software as well as advice on how to do things. We have recently become very interested in the NASA Jet Propulsion Labaoratory and their files of space photographs. We have found two versions of a program called "Image" that allow us to analyze these photos.

A new service that was initiated this year on MAGNUS is a menu driven "OASIS" (OSU Academic Services Information System) program that allows us to look at current news and weather, use the WHOIS function to find addresses and read online newsletters and journals. Our primary use this year has been KidsNet and the Hypercard discussion lists. By reading the lists regularly, we keep up with information from Apple, Apple's engineering staff and technologists who monitor the newsgroup. KidsNet is located at the University of Pittsburgh and monitored by Odd dePresno in Norway. The group mails a newsletter monthly in addition to providing a forum for educators to exchange views and research. The network also has a section for children called KIDS-92.

MAGNUS and INTERNET are more difficult for the students to master. They struggle with the complex addressing system, commands in UNIX and the different message handlers for the newsgroups. Several book length manuscripts have appeared this year which describe how to get the most from INTERNET. Once they pick it up, however, they become avid users in short order. The students can connect to Learning Link and AppleLink by way of gateways from INTERNET.

AppleLink and Apple Global Education

The network that we have been most active on is Apple Global Education (AGE). AGE is currently in over 40 countries with over 250 schools. The three major projects that I will report on come from AGE and are the: 1) Earth Day Celebration Newspaper, 2) First Nation Project and 3) This Week in AGE.

The Earth Day project grew out of a common interest among 30+ schools in communicating their concerns for the environment. Young children all over the world are interested in the environment. We set the project up so that the children would start out as "reporters" for a newspaper. In this role, the UCOT staff and students were paired with students at Marigold Elementary School in Victoria, B.C. Each group chose a topic from a list and proceeded to develop the topic, much as a reporter would. Personally, I was paired with two students who were to work on "drift net fishing."

In the first phase of the project, the two students, Darren Pendergast and Garrett Stevenson, sent me photographs of themselves and short biographies so that I could get to know them. I responded with my photograph and a biography. This was a very important piece of the correspondence puzzle, as we began asking each other questions about things we read in each other's biographies. I was very impressed with the quality of communications between Garrett, Darren and myself.

In the next phase, the two boys sent me an outline of their project as they had developed it. I looked it over, made some brief critical remarks and sent it back. I then proceeded to look up information in our library on the topic. I was able to locate numerous articles that I scanned and sent out to the boys. I

also scanned several charts and graphs which helped them do their end of the project. As the project picked up, several of our faculty members became quite interested in the project and wrote long essays to send to the students.

It should be pointed out that the UCOT-Marigold association was only one which was taking place in over 30 locations. We mailed rough drafts back and forth for about two weeks, commenting and rewriting the drafts as we went. In the last phase, the children played the role of newspaper editor. All of our work went to Tony Scott at the University of California at San Diego. On Earth Day, he mailed all the articles back to all of us at each location much as the Associated Press. We downloaded the articles and put together our newspapers complete with graphic illustrations, photographs, articles and essays. I was most impressed with the quality of the writing that came from the children. It should be pointed out that they were sixth graders at the time.

The First Nation project is another project between UCOT and Marigold Elementary in Victoria, B. C. The principal at Marigold, Dave Allan, had once been the principal of the Craigflower Elementary School, also in Victoria, not far from Marigold. The school is of interest to Dave and the other Canadian schools because it is approximately 80% Songhee Indian in population. As most are aware, the Native Canadians have a history that is similar to the Native Americans and many are related. The closeness of the school provided an excellent opportunity for the children at Marigold to become intimately acquainted with the Songhee nation and people, firsthand. In addition to UCOT, the Open School in Los Angeles, California and Rovaer Skole in Norway are participating in the project.

Again, at OSU we served as mentors and participants in the project. Our first assignment was to digitize photographs of ourselves to send to Marigold. We then wrote biographies to accompany the photographs. We digitized our voices saying our name and a short greeting. The next task was to go through our picture albums and find family photographs that meant something to us and write stories around the themes "The Story My Picture Tells," and "It Happened to Me."

I would interject here that it is projects like these that encourage students (and faculty) to learn how to use technology in a number of ways. It was an excellent project for our beginners because it gave them something to write about (word processing). It is difficult to pick a picture from your photo album that means something to you because you have to have some method for choosing, at least in my case, when my family saw me poking in the album, they began asking questions and before long, five more people were involved in the project. In my case, my four children ask a lot of questions and so I had to relive many situations in our family before I came up with one choice. This caused our family to learn a lot about each other. After we sent our pictures (using a digitizer and graphic manipulation software) and stories, we were eager to read the files from the other children at their schools and find out more about them. Of course, this stimulated more questions and caused more interaction to occur.

Again, I am most impressed with the writing that occurs from sixth grade students. Such a project incurs much discussion and lots of communications that require writing and careful thought. At the time of this writing, the riots are still ongoing in Los Angeles surrounding the Rodney King affair and we are anxiously awaiting news from the Open School. Here are some early comments...

"Everyone in our school community is very challenged by what happened. Many were directly effected by the fires and looting. One child lost a 15-year-old cousin. Another family hadn't had anything to eat for a few days, so we gave them a large cabbage and one beet from the garden, the only plants that were mature enough to pick. Our students who were not directly in the path of harm's way could still relate empathetically with their many classmates who were directly effected and so in this sense, the impact of the rebellion reached us all."

"The youngest students needed much adult attention. Their anxieties tended to spill over in overreaction to unintentional bumps from classmates. They needed explanations and reassuring hugs from the adults much more than usual. Their drawings portrayed darkened clouds and darkened suns or partially hidden

suns. But when gardening with me, they were exceptionally productive. The 6- and 7-year-olds managed one morning to double-dig four rows, plant 2 tomato bushes, 2 Japanese eggplants, plant six pots of aromatic herbs and aloe, and, another morning, weeded four rows of corn and carrots!"

"The older students have been writing about the disorder. I have read only what my AGE kids have written, which we sent to TWA. One girl, Iesha, had been especially withdrawn until we began discussing what they might write about. Her family had had to move in temporarily with an aunt. For most of the Open School folks, the worst of the protesters' acting out had stopped by Friday, May 1st, but for Iesha, the disturbances were still going on Sunday night in her South Central neighborhood. Iesha decided to send This Week in AGE an eye-witness account of what had happened in South Central, so she interviewed two of her friends living there."

"In my neighborhood, Koreatown, the store owners have already begun replacing shattered windows and restocking what was looted (in my block, most of the looting was done by El Salvodorans, who took food, clothing and shoes). Businesses that were set afire all burned to the ground. During the daytime, a normal routine has returned. The Latino produce and ice cream vendors are back on the streets, jingling their bells or broadcasting festive, attention-getting music. The Afro-American safety patrol lady has returned to guide the Korean and Latino school kids across the street. Drivers have resumed the cheery custom of announcing their arrival at a friend's apartment building by honking their car horns specially tuned to play a wide-interval melody with a similar contour to "La Cucaracha." But at night, I still hear more than the usual amount of gunfire, helicopters and sirens, although this also is tapering off. I don't know when peace and clarity will be fully restored. The potential for more violent confrontations in the future seems very real, especially where I am living."

As I write, Dave and the kids are finishing up the first set of stacks describing the project and are beginning to put together videotaped interview data for inclusion in their QuickTime movies and videodisk about the project. I hope to have examples of the work to share with this group in June.

AppleShare Remote

In our latest project we are trying to get our system to work with AppleShare Remote, Apple's proprietary system that allows the user to connect to a host computer from a remote site simply by typing in the phone number and password if needed. It allows me to connect to my local area network at my office from my home and leave files or messages for students. I can also see the file server where their files are stored. With this capability, I can scan their folders for homework, read it, grade it, if necessary and keep records in my own file from the convenience of my own home.

I can also demonstrate the use of local and wide area networks to teachers in the field directly from their own telephones. Currently, our router allows the use of MAGNUS and the INTERNET from a remote site across AppleShare remote as well as access to the local network. We will use this capability to send in articles to "This Day in AGE," the weekly newsletter on Apple Global Education. This newsletter grew out of the Earth Day project and has continued in operation for two years. The most amazing part of this project is that the articles are mostly submitted by children. Editorship of the newspaper is shared among AGE schools on a rotating monthly basis.

References

Hunter, Beverly. SUMMARY OF COMMENTS FOR THE CONGRESSIONAL STAFF FORUM ON "DESIGNING AN EDUCATIONAL TECHNOLOGY INFRASTRUCTURE: DEFINING THE FEDERAL ROLE IN THE DISSEMINATION OF INFORMATION." Internet, April 2, 1992.

Hunter, Beverly (1992). "Linking for Learning:Computer-and-Communications Network Support for Nationwide Innovation in Education". J. Science Education and Technology, Vol. 1 No. 1 pp 23-33.

Meeting the CHALLENGE: Providing voice, data, and video communications to local primary and secondary schools.

Alessandro C. Calianese, Ph.D.
Director, Academic Computer Center
Professor of Computer Science
St. Peter's College
Jersey City, N.J. 07307
Calianese_a@SPCVXA.SPC.EDU

Abstract:
In 1989 Saint Peter's College won a $3M grant sponsored by the State of New Jersey to develop an Institute for the Advancement of Urban Education. Ten local schools, five primary and five secondary, were invited to participate. The purpose of the institute was to develop, implement, and evaluate models of quality pre-college and college urban education with national application potential. An electronic bulletin board (EBB) and telecommunications network (TCN) were developed to provide an infrastructure linking all project components with each other and with universities around the country. This paper focuses on the planning, design, implementation, services, and uses of the EBB by the project components. the design, implementation, and uses of the TCN will be discussed.

In 1989 Saint Peter's College applied for and was awarded a three year grant for $3,270,702 from the State of New Jersey under the Governor's Challenge to the Independent College Grant Programs. It was a competitive program that awarded only two grants.

The Grant Narrative explains its operation over the past 3 years. Following the narrative is an explanation of the support the Electronic Bulletin Board and Telecommunications Network provided to the grant.

INTRODUCTION

Saint Peter's College is an independent, coeducational, liberal arts college serving 1,700 students and 1,400 adult learners from a multi-cultural, urban environment. Today, as over a century ago, the College is a gateway of opportunity for successive waves of new Americans and for the adult learner and at-risk college student.

Saint Peter's College brings to this mission educational principles which have characterized the four hundred year-long tradition of Jesuit education: personal care, academic rigor, and community concern.

Saint Peter's College responded to the Governor's Challenge to Independent Colleges and Universities by drawing on its own areas of strength to address shared problems with other Colleges in the quest for undergraduate education. The College created an Institute for the Advancement of Urban Education (IAUE) in order to develop, implement, and evaluate new ways of working with a multi-racial, multi-ethnic urban student body from varied socio-economic backgrounds. IAUE was designed to develop and apply models of quality college and pre-college education with potential for national application. The College actualizes its educational principles in an institutionwide system that embodies its Jesuit tradition and provides the nurturing support, educational skills, and experience of community so critical to the academic development of a heterogeneous urban student population.

A. Objectives

IAUE is devoted to the continuing study and improvement of college and pre-college education of an urban student population diverse in both academic skills and social background. The goal is the development of graduates possessing those qualities which have always been the hallmarks of Jesuit higher education: the ability to write and speak with clarity, to analyze and synthesize knowledge and experience, and to make value judgements in the context of a well developed sense of ethics. The curricular renewal central to this grant was designed to achieve the following objectives.

- To improve the pre-college preparation and motivation of urban, traditional and nontraditional students;
- To provide effective access to quality college education for the spectrum of urban youth, from the at-risk to the talented;
- To stimulate, reward, and retain students of a range of preparedness from matriculation to graduation;
- To achieve high standards of personal character and academic excellence among graduates.

Saint Peter's believes that these objectives can be realized by an implementary strategy which encompasses several key elements.

- Curriculum Action Team, led by Visiting Teaching Scholars, will create curriculum development networks among area elementary and secondary educators, within the College, and among other universities and colleges around the nation;
- Communications and reasoning skills will become integrated throughout the College's curriculum forming the bedrock of the liberal arts program;
- Educational enrichment opportunities will be made available to students from kindergarten through college;

See **Figure 1 - IAUE Organizational Chart**

B. Activities and Schedules

The cornerstone of all the program activities encompassed by Saint Peter's IAUE is the visiting Teaching Scholars program. Outstanding academic professionals, scholars and skilled teachers, are devoting time over several months a year providing the leadership for a curriculum development project that reaches from kindergarten through college.

These Visiting Teaching Scholars are drawn from five traditional academic disciplines -- natural sciences, mathematics, social sciences, business, and history; one interdisciplinary field -- international and intercultural studies; and three skills areas--written and oral communications, critical thinking, and computer-assisted instruction. The Visiting Teaching Scholars will lead Curriculum Action Teams in each of the fields identified, seeking solutions to the pedagogical problems posed by today's urban student Curriculum Action Teams are composed of Junior Scholars, drawn from the College's most gifted students, and Institute Fellows drawn from Saint Peter's faculty and senior teachers from New Jersey elementary schools and secondary schools with large, minority student bodies. The nine Visiting Teaching Scholars, ten College faculty members, ten Junior Scholars, and twelve pre-college teachers constitute the Institute's driving

force for the development of curricula characterized by an ease of academic access, excellence of educational result, and concern for values development. This collaboration is expected to produce significant curriculum enrichment and improvement.

IAUE conducts demonstration models of enriched curricula, infused with an emphasis on skills study and interdisciplinary/intercultural study, in the five traditional disciplines on both elementary and secondary levels. The participants in this demonstration project will include the Institute Fellows joined by thirty-six K-12 teachers, and thirty-six college students who will assist those teachers in the classroom for five hours a week for thirty weeks. **An electronic bulletin board,** described below, will provide an essential communications link among the participants.

On the Saint Peter's College campus, IAUE conducts Summer College Curriculum Infusion Program (CCIP), to make oral and written communications skills, critical thinking, computer-assisted instruction, and international/inter-cultural concerns integral to every component of the college curriculum. This program will be led by the Curriculum Action Teams discussed above, joined by the fifteen Saint Peter's College faculty each year.

The IAUE has convened two Annual Curriculum Education Symposia to share findings, theories, and ideas with other educators and researchers in related fields of interest. Other Scholars are invited each year to participate along with the Visiting Teaching Scholars, Saint Peter's College faculty and administration, Institute Fellows, Student Scholars and guests. The proceedings of these symposia will be aired over a **telecommunications network** described below, and subsequently published.

IAUE is forming partnerships among College faculty, area pre-college teachers, both public and private, and college students to provide for that preparation for college which should begin in kindergarten. This partnership program pairs College students with elementary and secondary teachers and places them in the schoolrooms for a full year. Students meet regularly with appropriate College faculty to discuss problems, methods, and assignments. The resulting network of education professionals created by IAUE programs in elementary and secondary schools facilitates achievement of one of Saint Peter's prime objectives: to identify talented students of all ethnic backgrounds and to encourage them to pursue higher education.

To supplement well-established and respected programs, the IAUE will develop a Pre-College Basic Skills Program for high school seniors needing to strengthen their writing, reading, and mathematical skills before college entry. The program, to be designed and taught by high school students and will operate from January to August outside of regular school hours.

Non Traditional students in Saint Peter's Evening and Weekend Sessions also will benefit from the curriculum development, and, in addition, the Saint Peter's Adult Reeducation Center (SPARC) will be established to provide basic skills training to adults. Completion of this year-long program in reading, writing, and mathematics will enable the student to continue in any College degree program.

Important functional elements in the Institute's program are an Electronic Bulletin Board (EEB) and a Teleconferencing Network (TCN). Together they constitute a communications and outreach structure linking all the components of the program to all its venues and beyond.

The EBB, operating through telephone lines to computer terminals, provides the basic information system. Accessible by participating schools and at many points on the Saint Peter's College campus, the EBB provides an electronic mail system and message board for all participants. General and program data and curricular development materials are transmitted by its means. This technology facilitates conferencing among team members in the development of

computer-assisted instruction programs. It also allows IAUE to contribute to national and international educational data banks and bulletin boards through linkages like BITNET, Internet, and USENET as well as to access their resources.

The Telecommunications Network provides for the Institute's more nuanced communications needs, when voices and visuals are important. IAUE's system provides one-way video and two-way voice and data between Saint Peter's College and any of the downlink sites and among local systems linked by telephone line. The TCN allows the Institute to provide simultaneous instruction in diverse locations -- participating schools, remote instruction sites, viewing centers for other interested educators -- and to make short professional presentations to many sites without interfering with normal schedules.

The EBB and TCN are pivotal in the Institute's program strategy. They are key to curricular development, to program implementation, to outreach, and to dissemination. They encourage information exchange and offer an economical, convenient alternative to frequent travel, extensive release time, and telephone contact which in large schools systems can be difficult and unreliable. They represent a form of literacy which the College wishes to encourage in all its constituencies, and they offer an opportunity to develop a demonstration project in the academic use of electronic communications technology and techniques.

See **Figure 2 - IAUE Operational Structure**

The general timetable for all these program follows. The first year was devoted initially to element definition and design (including project organization, planning, research, and curriculum development) and later to the initiation of programs, the second year the original elements were tested and additional ones introduced, the third year (in progress) is devoted to refining and extending the elements and evaluating the success of the project.

-Year I: Participating schools are announced and the Curriculum Action Teams formed; installation begins on the EBB/TCN system. Curriculum development begins and the Pre-College Basic Skills Program is initiated; the first of the Symposia and summer Curriculum Infusion seminars take place.

-Year II: Pilot curricula are now in participating schools and the College; the SPARC program begins in the College Evening and Weekend Sessions; all word processing equipment and software are in place. The pilot curricula and other program elements introduced thus far undergo refinement; all EBB and TCN capabilities become fully operable. The annual Curriculum Education Symposium and the Curriculum Integration Program are again held.

-Year III: The Pre-College Curriculum Infusion Program expands to include an additional thirty-six teachers, and the College Curriculum Infusion Program brings in another fifteen College faculty. Mid-course correction and program refinement are conducted. Major educational outreach efforts will continue throughout the year, including dissemination of research results. The Symposium and summer CIP end the year.

More information regarding scheduling is contained in the discussion of project hiring Section 4 in the Budget Narrative. Of course, project evaluation and research dissemination are both ongoing aspects of the project.

C. Outcomes and Benefits

Saint Peter's College looks with confidence to the IAUE as the vehicle for producing many desired benefits. Some of these are general future outcomes about which the Institute can only conjecture ; others are sufficiently measurable and near-term for the Institute to assess them.

Immediate Outcomes

During the three-year project period, the Institute will:

- Involve as many as 75 Saint Peter's faculty -- 60% of its full time members -- in an educational initiative with potential national importance.
- Offer a training opportunity for 80 -100 elementary and secondary teachers;
- Give up to 30 Saint Peter's College students an opportunity to test their interest in education careers by engaging in substantive, professional research and another 72 by assisting pre-college teachers in the classroom.
- Create supportive relationships and networks among the faculty and students of the College and the area's public and private elementary and secondary schools;
- Conduct applications research on existing models of instruction for improving the education of urban student populations;
- Develop new models of instruction for these students;
- Advance applications of critical thinking and computer-assisted instruction to curriculum development;
- Increase College and pre-college faculty and improve subject knowledge and teaching skills for pre-college teachers;
- Develop interdisciplinary approaches to education on all academic levels;
- Implement redesigned curricula at Saint Peter's College and all participating grade and high schools;
- Create and implement an innovative model of academic outreach to the community;
- Prepare students for participation in an international, multicultural society;
- Model academic uses of modern communications technology to enhance curriculum development, instruction, outreach, and dissemination.

Long-range Benefits

The College believes that the program developed by IAUE will contribute to:

- Improvement in individual and community quality of life in urban areas through increased intercultural understanding;
- Improved communications and reasoning skills among students from kindergarten through college;
- Better preparation of high school and college students of all ages for careers and career changes;
- Better retention of talented students within the State system of higher education.
- Enhancement of New Jersey's reputation as a national center for quality educational opportunity.

EBB AND TCN SUPPORT

Ten local schools were selected to participate in the IAUE. All were located in Hudson County in the State of New Jersey. All were within a 10 mile radius of the main campus of Saint Peter's College in Jersey City. Four were primary schools and five were secondary schools. Two additional schools were added to the grant in academic year 1991-92 (one primary and one secondary).

SCHOOL	P/S	CITY
1. Bayonne H.S.	S	Bayonne
2. Dickinson H.S.	S	Jersey City
3. Ferris H.S.*	S	Jersey City
4. Hoboken H.S.	S	Hoboken
5. Hudson Catholic H.S.	S	Jersey City
6. Kearny School	P	Hoboken
7. Kennedy School	P	Jersey City
8. Lincoln School*	P	Bayonne
9. Memorial H.S.	S	West New York
10. North Bergen	S	North Bergen
11. St. Patrick's School	P	Jersey City
12. Wakeman School	P	Jersey City

* - Added in Academic Year 1991-92

EBB EQUIPMENT

SCHOOL SITE EQUIPMENT

The equipment installed at each school is identical. The configuration at each site is:

* IBM PS/2 Mod 25 with 20 Mb hard drive and a 2400 Baud Modem
* Okidata 195 printer
* A single private telephone line not tied into the internal Telco system.
* System locking security device, glued version. The device secured system, key board and printer.

Installation at each site proceeded smoothly. At some of the sites power had to be re-installed.

SAINT PETER'S COLLEGE ACADEMIC COMPUTER CENTER EQUIPMENT

The portion of the Academic Computer Center's VAX cluster that handles the EBB are:

* DEX VAX 8650 64 Mb, 3Gb disk running VMs
* DEC VAX 11/785 32Mb sharing disk storage with the VAX 8650 over a VAXcluster

The following equipment was installed to support the EBB portion of the Challenge Grant:

* 15 modems and phone lines at SPC. These do 2400-9600 baud.

* Terminal server cards (2) model CXY08 for connecting the modems.

Software:

* MS-DOS Kermit with local customizations on the 25's.

* VMS Kermit on the VAX.

* VAX Notes conferencing software from DEC.

* ANU News conferencing software.

SOFTWARE SERVICE SUPPORT

The approved grant required only the development and maintenance of a local EBB. The software package chosen was a DEC product called NOTES. A dedicated conference (CHALLENGE) which handles messages grouped into 36 overall topics was established for the participants.

It was clear early in the grant that it would be desirable to communicate with colleagues outside our restricted EBB. Saint Peter's College purchased a DEC product called News.

News is the replacement for the NOTES system. While NOTES are private to SPC, News messages go around the world to millions of readers (although not all readers read all messages). We currently have 127,412 messages in the News system, grouped into 1,198 discussion groups. Of particular interest to us are the K12.* newsgroups, which discuss various issues in the pre-college school system.

K!Z.* NEWSGROUP (LIMITED EXAMPLE LIST):

k12.chat.elementary	Informal discussion among elementary students, grades K-12.
k12,chat.junior	Informal discussion among students in grades 6 - 8.
k12.chat.senior	Informal discussion among high school students.
k12.chat.teacher	Informal discussion among teachers in grades K-12.
k12.ed.art	Art curriculum in K-12 education.
k12.ed.business	Business education curriculum in grades K -12.
k12.ed.comp.literacy	Teaching computer literacy in grades K-12.
k12.ed.health-pe	Health and Physical Education curriculum in grades K-12
k12.ed.life skills	Home Economics and Career education in grades K-12.
k12.ed.math	Mathematics curriculum in k-12 education.

k12.ed.music	Music and Performing Arts curriculum in K-12 education.
k12.ed.science	Science curriculum in k-12 education.
k12.ed.soc-studies	Social Studies and History curriculum in K-12 education
k12.ed.special	K-12 education for students w/handicaps or special needs.
k12.ed.tag	K-12 education for talented and gifted students.
k12.ed.tech	Industrial Arts curriculum in K-12 education.
k12.lang.art	Language Arts curriculum in K-12 education.
k12.lang.deutsch-eng	Bilingual German/English practice with native speakers.
k12.lang.esp-eng	Bilingual Spanish/English practice with native speakers.
k12.lang.francais	Bilingual French/English practice with native speakers.

Other services available to CHALLENGE participants are:

BITNET:

Saint Peter's College participates in BITNET, an international e- mail service. All participants have access to this service.

INTERNET:

Our Internet connection allows users to exchange mail with anyone who is connected to the network. There are hundreds of thousands of computer systems worldwide to which we are linked. These other users can also access our computer system, and we can access theirs, for purposes of sharing useful files e.g. public-domain educational packages, hypercard stacks for MACs, etc. There is also a real-time "chat" capability.

PRODIGY:

The Prodigy is a personal computer service that allows a user to interactively:

* See the latest news and sports
* Check local, regional and national weather
* Book airline reservations with EAASY SABRE
* Order groceries delivered to your home (in some areas)
* Get advice from nationally known experts
* Shop from dozens of different merchants
* Get stock quotes from Dow Jones News/Retrieval
* Buy and sell stock from a discount broker
* Send and receive electronic mail
* Tune in to the latest entertainment news
* Learn from many educational features
* Play challenging and fun games
* Look up product information in Consumer Reports

Currently there are 229 users registered on our VAX for CHALLENGE.

TCN EQUIPMENT

The Institute provides a channel for live communications between each member school. A renovated television studio at Saint Peter's College is used for this communication. Television programming, originating from the College, is sent by satellite to all schools and feedback and questions are routed by an interactive telephone link from each school to the television control room.

Each site received a system comprised of a polar mount, 3.7M(12') solid fiberglass dish with a Chapparal CoRotor C and Ku feedhorn into a Chapparal model 30 dual C/Ku receiver. The size of the dish was determined by the minimum acceptable size for adequate reception of half-transponder Ku satellite transmissions. One exception was the final site, the combined grammar and high school, which received a 3.4M dish because of the increased gain of the newer dishes. As most schools requested a roof installation for security reasons, most mounts employed the rugged Baird PXL-2 mount in a penetrating configuration. All bases were designed by a structural engineer to withstand the 110 mph survival rate for the dish.

The link between the College and our uplinker, Cresscom Transmission Services, is rented from New Jersey Bell and employs a three-hop microwave link between Jersey City and the uplink earth station at Holmdel, New Jersey. Audio is routed through a Tascam M-208 audio console and telephones are handled with a Symetrix 104A Multi line telephone interface.

We began with full transponder Ku telecasts with added C-band coverage for our nationally advertised symposiums. Recently we have switched to C-band satellites and are currently using Galaxy VI for our telecasts.

See **Figure 3 - TCN Structure**

TCN PROJECTS

To date three local telecommunication series and two national satellite telecasts have been completed and one additional series scheduled for May and June 1992. All telecasts are live transmissions from a TCN studio at
St. Peter's College. All telecasts run 45 minutes. The following is a list of topics and dates of the telecasts:

VIDEO SERIES IN METHODS AND CURRICULUM FOR THE URBAN CLASSROOM

* Introduction to IAUE Comments on the Quality Education Act Introduction of local schools The Future of IAUE	-	October 31, 1990
* Critical Thinking Team	-	November 14, 1990
* Computer Assisted Introduction	-	November 28, 1990
* Mathematics Team	-	December 5, 1990

* Oral and Written Communication - December 12, 1990
* History - December 19, 1990

VIDEO SERIES IN CURRICULUM ENRICHMENT FOR STUDENTS K-12 IN SCIENCE, MATHEMATICS, MULTICULTURALISM, AND WHOLE LANGUAGE

* Multiculturalism: Today's Classroom - January 14, 1991
* Mathematics and Co-operative Learning - February 4, 1991
* Mathematics and Classroom Anxiety - February 11, 1991
* Science: Experiments and the Environment - 1 - March 3, 1991
* Whole Language - March 24, 1991

VIDEO SERIES ISSUES AND SOLUTIONS IN URBAN EDUCATION

* Youth Leadership - March 20, 1991
* Disaffected Youth - March 27, 1991
* Parental Relationships - April 10, 1991
* Child Abuse - April 27, 1991
* Substance Abuse - April 24, 1991
* Youth and Health - Part 1 - May 1, 1991
* Youth and Health - Part 11 - May 8, 1991
* School Restructuring
  - -Jersey City Take Over - May 15, 1991
  - -Site Based Management -
* School Administrators' Rountable - May 22, 1991

VIDEO SERIES IN PRACTICE AND POLICIES (SCHEDULED FOR MAY 1992)

* The New Students and Their Needs - May 11, 1992
* Schools, Violence and Discipline - May 18, 1992
* Health - May 26, 1992

(Note: Due to Memorial Day, this broadcast is on a Tuesday.)

* Vouchers and Urban Schools - June 1, 1992
* Humanistic Education - June 8, 1992
* City as a classroom - June 15, 1992

Two National satellite telecasts have been conducted from Saint Peter's College's uplink studio. Prominent television personalities hosted the telecasts. Both telecasts aired on C-BAND and KU-Band Satellites.

TEACHING STRATEGIES IN THE URBAN CLASSROOM INTO THE 21ST CENTURY K-COLLEGE

Moderator: **Rolland Smith,** Anchor, WWOR-TV

MODELS FOR THE 21ST CENTURY: PARTNERSHIPS, COLLABORATIONS AND URBAN EDUCATION

Moderator: **Gil Noble,** Emmy Award-winning host of WABC's "Like it is"

COMMITMENT BEYOND THE GRANT PERIOD

Because of the nature of the project, most of its programs are institutionalized simply by the adoption of the curriculum it develops. Other proposals may be put forward for its use. IAUE's annual costs after the three years of this project will be greatly reduced but remain substantial. Nevertheless, these costs will be incorporated into the College's operating budget. During the course of the project the College's Development Office will seek private funding to help support the continuation of these programs. Of course, the College will sustain its relationship with area primary and secondary schools and maintain the EBB/TCN infrastructure as part of its continuing outreach program under direction of the College's Director of High School and College Relations and the Director of Minority Affairs. The Pre-College Basic Skills Program would continue Saint Peter's Adult Reeducation Center. The curriculum developed for the College would be subject to the cyclical evaluation and outcomes assessment referred to above.

IAUE Organizational Chart

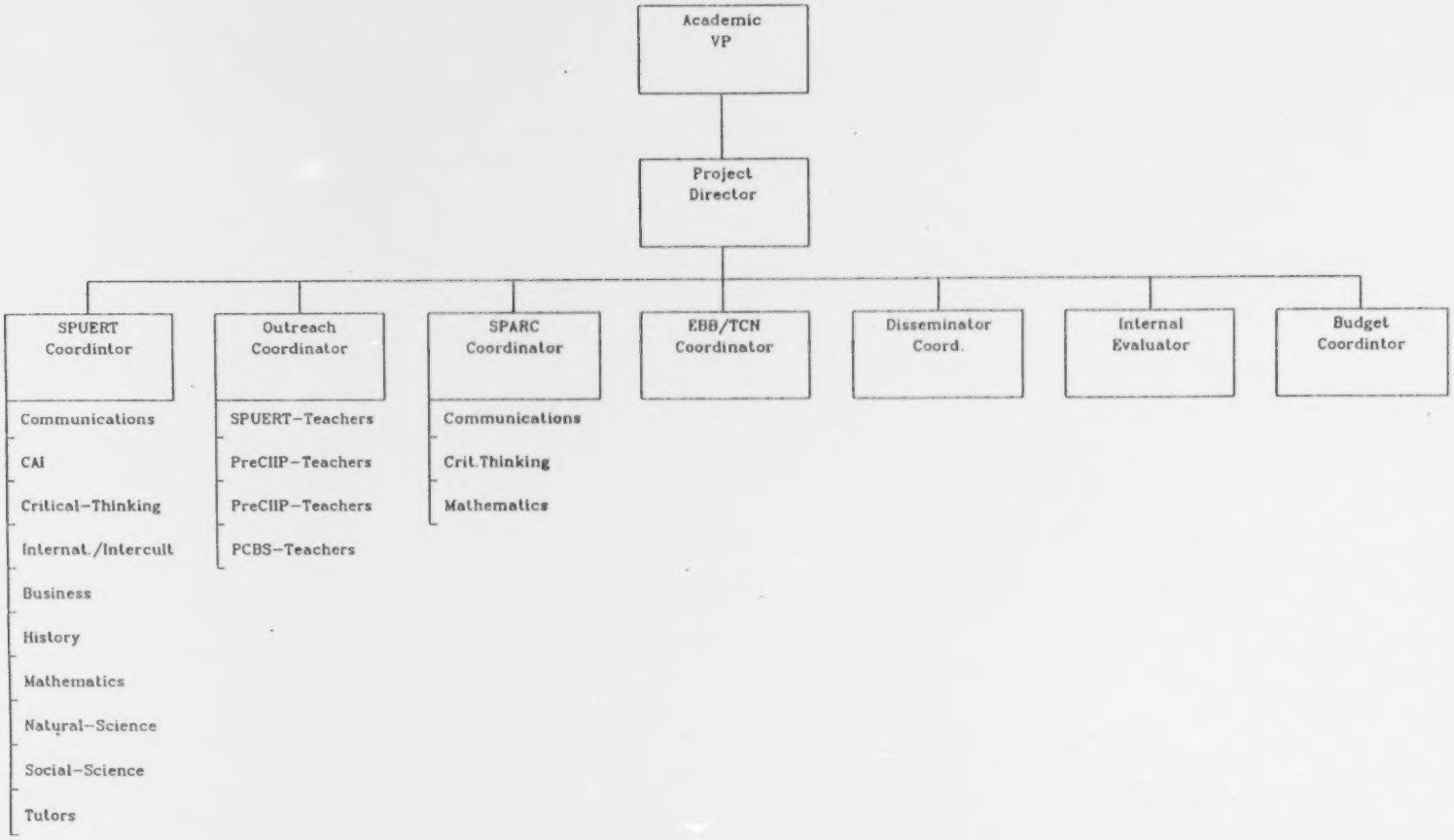


Figure 1

IAUE Operational Structure

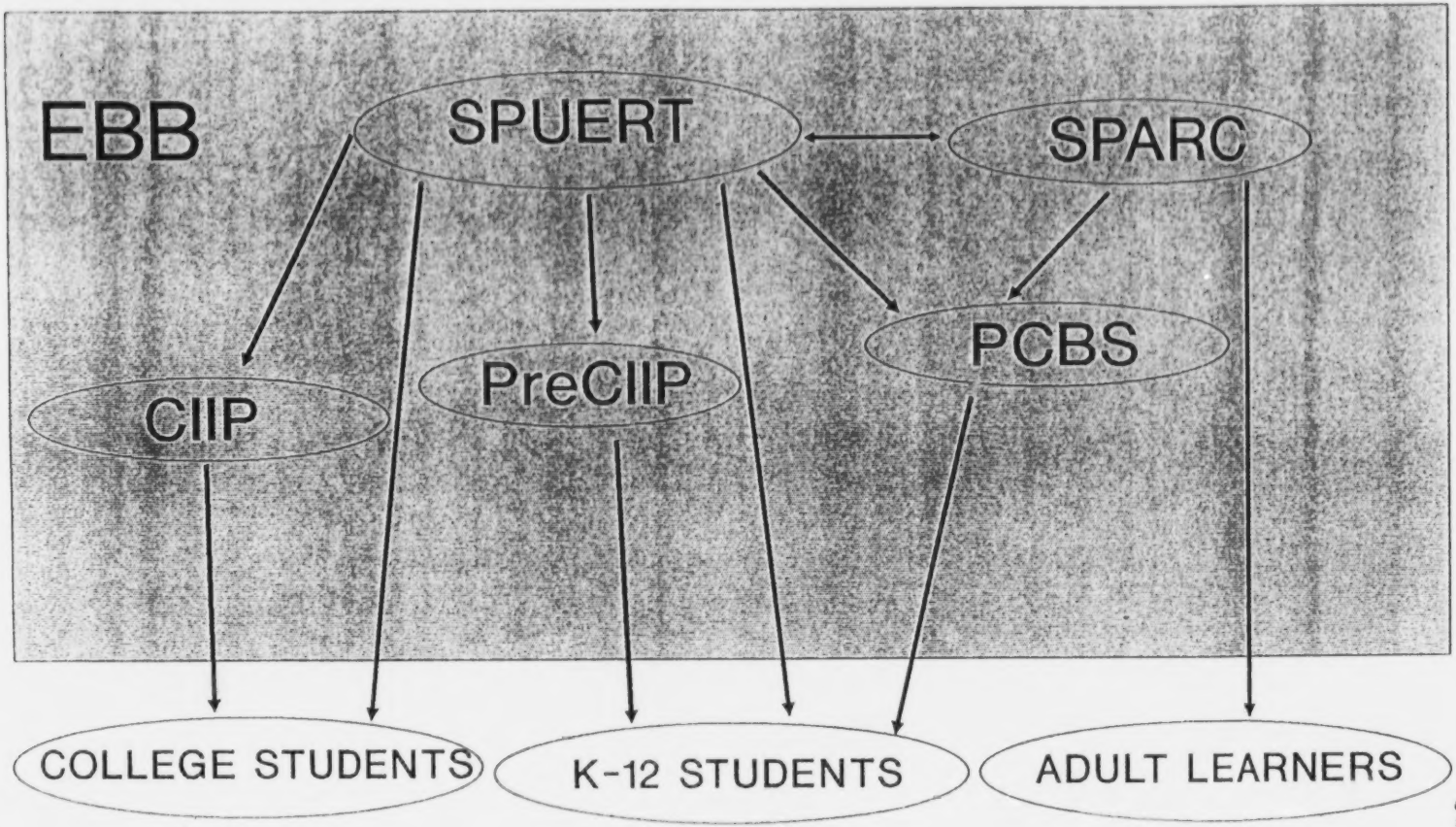


Figure 2

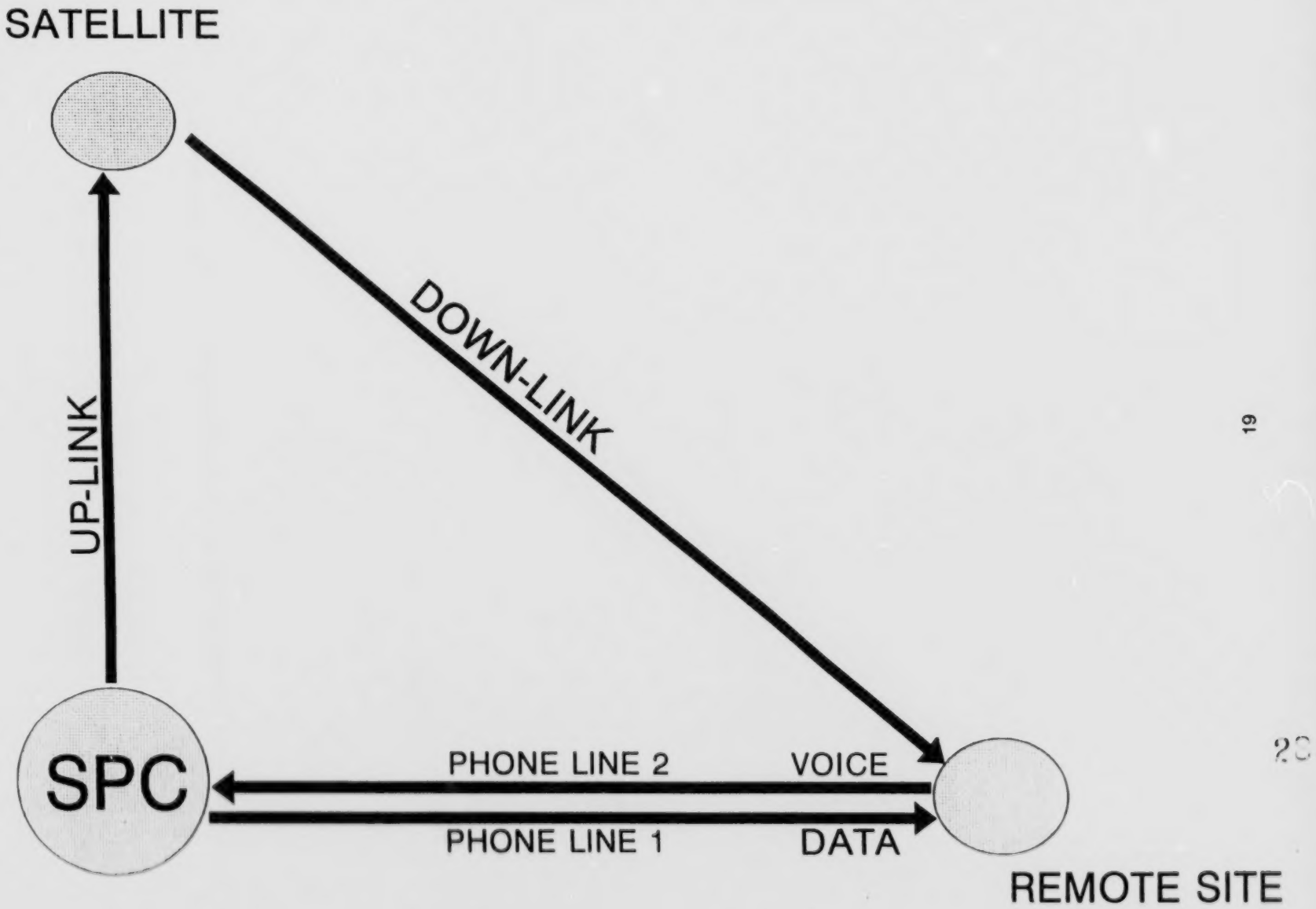

SATELLITE
UP-LINK
DOWN-LINK
SPC
PHONE LINE 2
VOICE
PHONE LINE 1
DATA
REMOTE SITE

Figure 3

Computers In Calculus
(and Throughout the Mathematics Curriculum)

Dr. G. Daniel Callon
Department of Mathematical Sciences
Franklin College
Franklin, IN 46131

The Department of Mathematical Sciences at Franklin College was chosen as one of the 101 Success Stories of Information Technology in Teaching and Learning in Higher Education in EDUCOM's Joe Wyatt Challenge. This paper will provide an overview of the program, its history, and its future. We will discuss our assessment of the benefits and drawbacks of our approach, both from a faculty and student perspective, as well as some unexpected outcomes. We will also analyze some of the factors in our success and to what degree they could be replicated at other colleges and universities.

Franklin College of Indiana is a small private four-year church-related liberal arts college founded in 1834. It was the first institution in Indiana to admit women and continues to strive to reach those students who are qualified academically but who might not otherwise have an opportunity to receive a college education. For example, 43% of this year's 909 students (843 full-time) are first-generation college students, many from small towns in Indiana. A strong financial aid program and increased retention rates have kept enrollments climbing in recent years, while the quality of students also has risen (our 1991-92 freshman class posted an average SAT score of 1000). A 12:1 student to faculty ratio allows Franklin to offer personal attention while maintaining high academic standards.

The Department of Mathematical Sciences, with six full-time faculty members, offers programs in mathematics and computer science. The department has a strong track record in preparing its majors for graduate school and for careers in education and industry. The department is also an integral part of the general education program of the college: every student is required to complete a mathematics course (either Calculus I, applied calculus, or a liberal arts mathematics class), and most are required to take a course in computer literacy and/or programming.

Our project evolved from our determination to teach mathematics more effectively throughout our curriculum. There has been a national movement in recent years to reform the way mathematics is taught at all levels, aimed at increasing student comprehension, appreciation, and motivation. For many students, so much energy is expended on practicing routine manipulations that such skills become the primary focus of study. Students begin to perceive mathematics as a disjointed series of hurdles to be conquered, with little cohesiveness as a field and little relationship to other disciplines or to daily life.

The increased availability of computer technology, both in terms of price and ease of student use, offered the mechanism for us to incorporate the principles of discovery learning in our classes to try to overcome this problem. Discovery learning shifts the focus of the classroom from the instructor to the student. Students are guided to discover and explore new concepts and to investigate their relationships with concepts already encountered, with the instructor serving as a resource in and out of the classroom. We have found that the principles of discovery learning work most effectively when blended with the ideas of cooperative learning, in which students are encouraged to work together in pairs or small groups while performing their investigations.

In 1989, Dr. Richard Park and I received separate $5000 summer stipends from the Lilly Endowment Inc. to revise the first two semesters of calculus into laboratory courses and secured permission from the academic dean of the college to offer the courses on a one-year experimental basis. Instead of meeting five hours each week for lecture sessions, the courses were scheduled for four hours of lecture and two hours of laboratory time in the campus computer lab, a 28-station PC lab open for class use on a first-come, first-served basis, and for student use at all other times. A menu-driven software

program called Quik-Calc, designed by Dr. John Buchanan of Miami University, was used exclusively in the first semester, with the symbolic manipulation program DERIVE introduced midway through the second semester. Reading assignments from current journals and books and writing assignments based on the readings were also added to provide as broad an introduction to the field of mathematics as possible. The experiment proved so successful that the change was made a permanent part of the curriculum.

Other changes rapidly followed. Dr. Paul Nugent received another Lilly summer stipend in 1990 to redesign the precalculus course along the same lines, although the laboratory exercises were not scheduled separately but incorporated into the general structure of the course. While computer exercises and discovery learning had previously been used in a limited way in several courses, their use quickly spread into the remaining lower-level classes and many of the upper-level offerings as well. Both applied calculus and the liberal arts math course began to use the computer as an important tool for graphics and for number-crunching. The use of DERIVE was extended to Calculus III to help students visualize three-dimensional graphs and to perform symbolic algebra manipulations. Courses such as differential equations, operations research, complex functions, and linear programming now all make use of specialized programs as integral parts of their curricula.

As this project has evolved over time, the vast majority of the work has been done by faculty members as part of their regular course preparations. The summer stipends were critical stimuli in making the necessary major changes. Besides the time investment by faculty in developing the applications for use in their courses, the use of discovery learning requires a great deal of time in dealing with students on a one-on-one basis both in and out of class. These time needs are ongoing, although the instructor does become more adept at working out the logistics after some experience. The major costs in human resources would come for institutions which have large classes; it would be difficult to hire graduate students or other support personnel with the expertise and interest to adequately serve as resource people under such a setup.

All our lower-level classes use the 28-station PC lab/classroom for their computer applications, as do most of the upper-level classes. This lab consists of Zenith 159's (IBM-XT clones) equipped with either VGA or EGA video cards. They are networked using DEC's Pathworks 4.0 and utilizing a MicroVAX II as file server. The lab also includes three sharable printers. The remaining classes use our 12-station VAX lab, providing terminal communications to the MicroVAX II. All full-time faculty members in our department have Zenith 386's in their offices as of 1991-92. The main factor in future planning is the ever-increasing usage of the lab by several departments, including our own. We will need to expand the facilities in some form in order to insure adequate access for both classes and for student use outside of class. The advantage of our program is that it can be adapted by any institution with adequate computer facilities for classroom use.

Currently plans are underway to expand the program in a major way. Funding is being sought to equip two new computerized classrooms, with priority for their use given to the mathematical sciences department. One would be equipped with 28 IBM-compatible 486's and designed so that students could have access to a computer on a regular basis (at least once weekly) during class sessions in lower-level courses in mathematics and computing. The second would run the software package Mathematica on 12-15 486 stations and would be designed for daily use by upper-level classes. Mathematica is a sophisticated package which not only performs graphing, numerical calculations, and symbolic manipulation, but also provides a computing environment with programming capabilities. The fundamentals of using Mathematica would be taught at the beginning of Calculus III, integrated into the usual review of material from Calculus I and II. Mathematica would then serve as a common thread throughout the upper-level mathematics program, with its notebook feature in particular providing students the opportunity to start with instructor-furnished examples and then to experiment with a large number of variations and changes. The department is currently operating under a $33,700 Curriculum Development grant from the Lilly Endowment, Inc., to provide faculty with adequate training and released time to begin to develop the necessary curriculum materials and applications.

Comparison of student test scores on class tests and final examinations has been encouraging as we have assessed the results of this project. Students demonstrate a much better grasp of fundamental concepts and are more adept problem solvers than those taught under the traditional format, without any loss of computational and manipulative skills. Formal class evaluations have documented improved student attitudes toward learning and increased faith in their abilities to work things out for themselves. Instructors also have noted that students are better at working independently, and yet seem to have developed more of a working relationship with their instructors and with their classmates.

We have been pleased at several offshoots of this approach to teaching mathematics. As a result of student initiatives, we have introduced two new courses, both optional and taught under the pass-fail format. In light of students' already full schedules, these courses earn one hour credit each by meeting for two hours weekly but assigning no homework. "A Mathematics Sampler," aimed at freshmen and sophomores, provides an overview of current areas of mathematical interest and topics from across the breadth of the mathematical spectrum for those who want to see where the mathematics they have studied might lead. This course was requested by students whose interest was piqued by the reading and writing assignments from the calculus sequence. The course is designed to involve students as much as possible in investigating the rudiments of a variety of topics ranging from nonEuclidean geometry to Markov chains to number theory, with lectures limited to establishing a foundation of definitions and basic results. Faculty members alternate the instructional duties. The second course provides upperclass students the chance to work in teams to prepare for mathematical competitions such as the Indiana state college mathematics contest. Each week a set of challenging problems is provided with students working together in teams of three to come up with responses as complete as possible. Student interest also continues to grow in attending research conferences and in pursuing individual research projects.

In comparing our efforts with those at other colleges and universities, we have noted several factors which have been crucial in the success of our work. The most important is that all our departmental faculty, both full- and part-time, have been committed to our goals and methods from the very beginning. In fact, the most senior faculty members have been among the most ardent advocates of using new approaches to improve student learning. This seems to be in marked contrast with the experience of many other departments. Secondly, by introducing any course changes to all sections at the same time, we minimized any negative feelings arising from the tendency of students to compare the amount of work they are required to expend with those of their friends taking the same course. This also brought the talents and perspectives of a broader group of faculty to bear on each course alteration. Finally, we are blessed with a creative, flexible, and enthusiastic faculty who are able to transfer their excitement to the students. No new approach can be successful without faculty with the ability and willingness to step away from the familiar techniques and launch out into the unknown, aware that there will be unforeseen problems and challenges. They must also be willing to devote the time necessary for such an undertaking.

The methods of teaching mathematics will be undergoing fundamental reforms in the next decade. An increasing awareness of the shortcomings of the current pedagogical techniques, coupled with the rapid developments in technology, have provided an impetus for change which is already making an impact. Mathematics educators have documented the substantial dropoff in the numbers of students taking mathematics courses each year throughout high school and college, and are beginning to shed some light on the reasons for their departures. The careful integration of computer technology can increase student interest, provide better and faster graphical and computational perspectives on concepts and applications, and allow students to bypass their shortcomings at mechanical skills to work at developing a global understanding of the mathematics they study. In fact, the fundamental principles, including discovery learning and cooperative learning through computer technology, can be put to use in any department for which computer applications are sufficiently developed. The size and scope of the integration can be adapted to take advantage of the computer resources available on a particular campus, so costs would be totally dependent on how much of the costs the individual department would be required to provide. If sufficient accessible equipment is already present on campus, the only costs for

a program on a scale similar to ours would be for software and for faculty time to revise courses. Most colleges which have made a start in this direction are willing to share their materials, so a total funding level of under $10,000 can sustain such a project for several years.

Given our experience, student and faculty reaction will be so positive that expansion of the project soon becomes not only desirable, but practically a necessity. This in turn creates new problems in regards to the availability of adequate computing resources. In our case, computer facilities went in three years from state of the art and extremely accessible to vastly overused and inadequate, despite the best efforts of computing center staff, faculty, and administration. This is where financial considerations can become a major factor, with budgetary constraints often forcing departments and colleges to look to outside funding for the necessary support.

In summation, we have found that using computer technology as a tool to foster discovery learning allows students to see mathematics as an experimental science and not as an exercise in memorization and example-matching. Such techniques produce independent learners who are more effective and self-motivated problem solvers, and yet more willing to perceive their instructors as resource aids in the learning process. By providing a variety of ways of perceiving new concepts, especially graphically and numerically in addition to analytically, computer technology has allowed us to better equip students with the necessary mathematical tools and the ability to use and adapt those tools in new situations.

Computer Modeling in the Traditional Classroom

Peter L. Havholm
Larry L. Stewart
Department of English
The College of Wooster
Wooster, OH 44691
PHavholm@Wooster
LStewart@Wooster

George P. Landow begins his recent book *Hypertext: The Convergence of Contemporary Critical Theory and Technology* (1992) by noting the parallels between computer hypertext and critical theories of intertextuality and deconstruction:

> When designers of computer software examine the pages of *Glas* or *Of Grammatology*, they encounter a digitalized, hypertextual Derrida; and when literary theorists examine *Literary Machines*, they encounter a deconstructionist or poststructuralist Nelson (2).

Our own work began with the recognition of this parallel when, in 1989, we taught a course in seventeenth-century British literature in which we asked students to use hypertext to create models of intertextual links among texts.

In thinking of intertextual links, we had in mind the literary theory of intertextuality as defined by Terry Eagleton: "All literary texts are woven out of other literary texts, not in the conventional sense that they bear the traces of 'influence' but in the more radical sense that every word, phrase or segment is a reworking of other writings which precede or surround the individual work." Our idea was that students would learn more both about seventeenth-century texts and culture and about the theory of intertextuality by attempting to create their own models than by traditional lectures and discussion or even than by studying someone else's models.

This course (titled "March 2, 1667" because we started with several texts associated in some way with that date) asked students first to consider what had to be available in the cultural to allow the formulation of a given text and then, using the hypertext program *Guide*, to create a model of the relationship between the given text and what we came to call the *underlying* texts, essentially to make a model of intertextuality. To use a very simple example: the book of "Genesis" had to be available in the culture to make possible the formulation of Milton's *Paradise Lost*. Thus, "Genesis" was an underlying text, and students using *Guide* might embed appropriate lines from "Genesis" under the relevant lines from *Paradise Lost*. A mouse click on a line from Milton's epic would bring to the computer monitor the underlying line from "Genesis."

However, as students soon came to realize, a line from "Genesis" was not enough to account for a line from *Paradise Lost*. There were commentaries on "Genesis" by Biblical scholars, visual representations of Adam, Eve and the serpent, poems in blank verse (a poetic convention necessary to the creation of *Paradise Lost*), and so on. In short, considering and modeling the intertextual relationships of even a single line helped students see the deep and rich ways in which texts were embedded and contextualised within their culture. It also provided the occasion for critically examining and discussing the theory of intertextuality which these students were modeling.

By the end of the course, we were convinced that students had learned a great deal about literature, culture and theory, perhaps more than they would have learned in a conventional course. Our perception was reinforced not only by enthusiastic student evaluations but by the testimony of English faculty, who found these students in subsequent courses to have a particularly sophisticated

understanding of textual relationships. The perceived success of the course suggested to us that we were on the right track: that the technology of the computer and hypertext allowed a kind of visualization of the theory otherwise impossible. The technology also allowed students to create their own models.

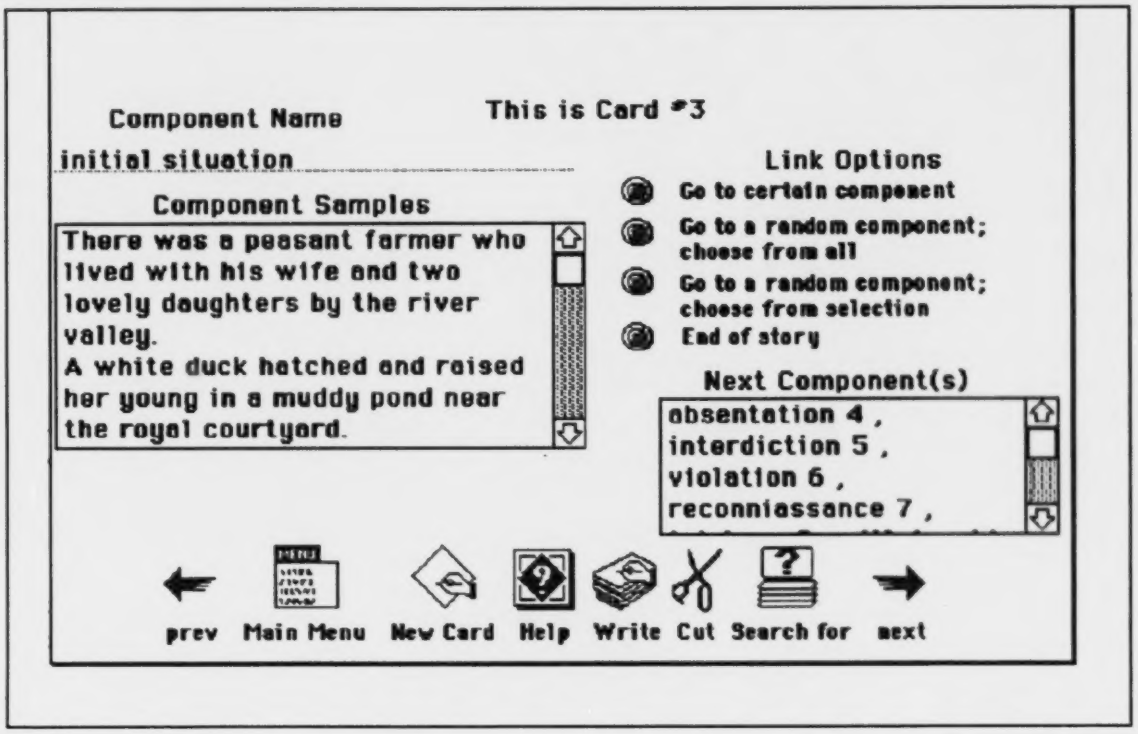
However, our question came to be whether technology gave us the means to model theories other than the intertextual. While the multidimensional metaphor of hypertext with its layers and links seemed perfectly suited to intertextuality, it seemed less helpful, for example, in modeling such linear theories as those of narrative. We turned then to *HyperCard* which allowed the creation of linear links in order to teach a course in which students modeled narrative theories.

In our first version of the course, "Story and Theory," we spent a substantial part of the semester simply discussing various theories of narrative, including Vladimir Propp's theory of folk-tale narrative and Aristotle's theory of tragedy. For the course project, we divided students into teams and assigned to each team a student-programmer, a person with a good working knowledge of *HyperCard* but not otherwise associated with the course. We then spent several class periods instructing all students in the use of *HyperCard*, not with the intention that they become expert users but in the belief that knowing something about the program would help them in the up-coming project.

The assignment was simply stated. Each group was to devise a *HyperCard* application that would produce skeletal narratives according to the theory of either Propp or Aristotle. The teams working with Propp would produce an application which generated folk-tale skeletons; those working with Aristotle would be able to generate skeletal versions of Aristotelian tragedies. Had either we or our students realized that we were asking the impossible of a group of undergraduates, most of whom were humanities majors with previous computer work limited to word-processing, the experiment surely would have failed. However, with our students painstakingly explaining the theory to the programmers and working with them to formalize the theory, each team succeeded in creating an application that in some way modeled the theory and produced narratives. The results of that class are detailed in "Modeling the Operation of Theory" (*Academic Computing*, March 1990). In summary, our conclusion was that though some of the applications were disappointing in themselves, that is, as narrative generators, making them had forced students to confront critical theory more precisely and thoroughly than occurred in most classes.

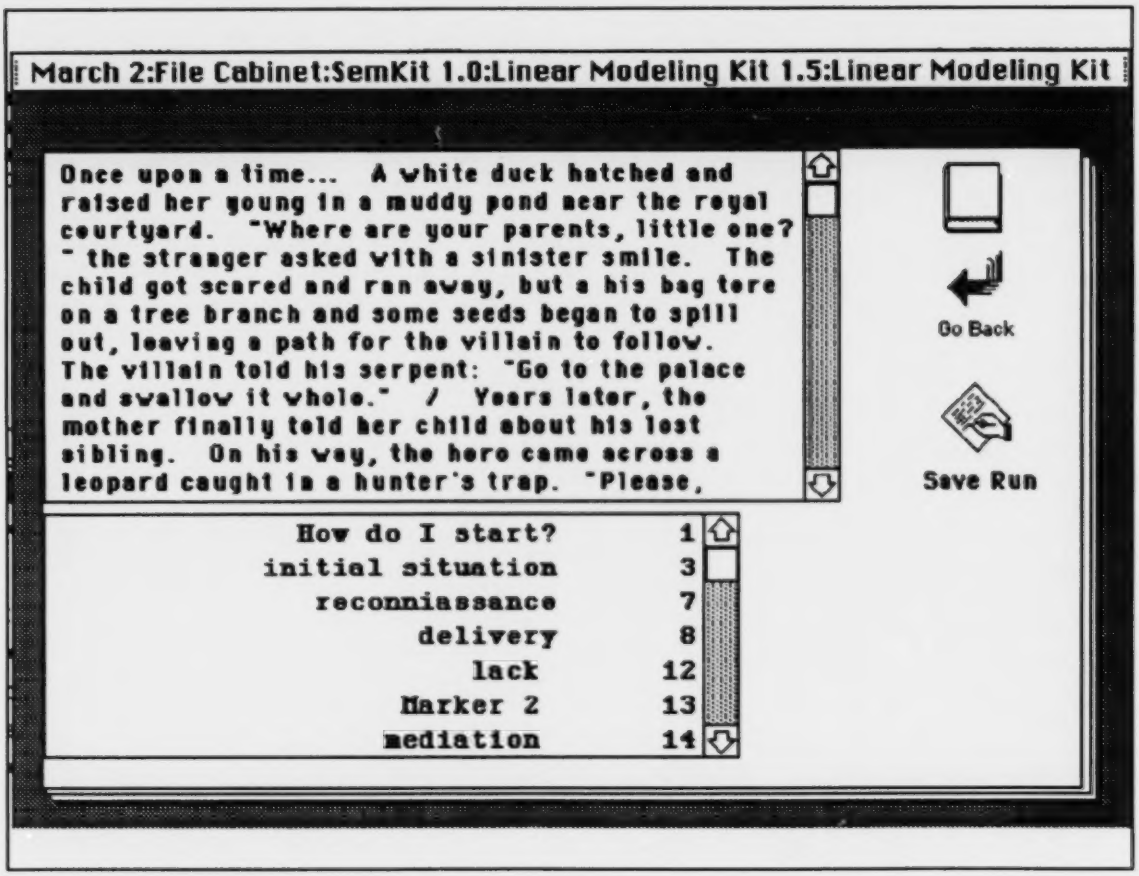
As encouraged as we were, we also recognized the problems of duplicating the project assignment in future versions of the course. We could not count each year on finding student-programmers able to give substantial amounts of time to the course and on getting funding to pay these programmers. Fortunately, the best of the student-produced applications gave hints of what would be needed in a more sophisticated narrative generator. During the next few months, again with the help of a student programmer, we devised a *HyperCard* application which was essentially a shell into which students could "program" their understanding of a narrative theory and then generate narratives. This application, which we first referred to as "The Linear Modeling Kit" and later as "The Syntactic Modeling Kit," allowed us in subsequent versions of the course to teach students to model narrative theory without the intermediary of programmers.

How the application works can be seen in the following example using Propp's narrative theory: Propp's analysis of certain Russian fairy tales argues that there are thirty-one elements (or "functions") and several combinatorial rules which account for all of the tales he studies. His claim, in other words, is that these tales can be constructed using only the elements and the combinatorial rules. The hypothesis invites testing.



Using the modeling kit, students can model the consequences of Propp's theory. A student has labeled the "card" shown "initial situation," one of Propp's 31 elements, and has invented actual narrative strings ("There was a peasant farmer . . . ," "A white duck hatched . . .") that fit the element's definition and could make stories.
The student has indicated which other elements the story may "jump to" from this one using the four Link Options at right. That is, one may instruct the narrative to go only to one other element, to any other element, or to one element from a specified group. In the case above, the user has instructed the story to jump to any of elements 4-7 and 9-12.

The "generator" uses these elements and rules to make stories. The resulting stories look like the following screen shot:


March 2:File Cabinet:SemKit 1.0:Linear Modeling Kit 1.5:Linear Modeling Kit
Once upon a time... A white duck hatched and raised her young in a muddy pond near the royal courtyard. "Where are your parents, little one? " the stranger asked with a sinister smile. The child got scared and ran away, but a his bag tore on a tree branch and some seeds began to spill out, leaving a path for the villain to follow. The villain told his serpent: "Go to the palace and swallow it whole." / Years later, the mother finally told her child about his lost sibling. On his way, the hero came across a leopard caught in a hunter's trap. "Please,
Go Back
Save Run
How do I start? 1
initial situation 3
reconniassance 7
delivery 8
lack 12
Marker 2 13
mediation 14

Note also that this card indicates which elements have been used to make this particular story. ("Reconnaissance," "delivery," etc. are Propp's terms.)

The class discussions which followed the students' making Proppian story generators went something like this.

It became clear immediately that one can generate a tremendous variety of element-combinations using Propp's simple rules. Further, allowing for the program's lack of semantic skill, most of its orderings of events were recognizable as possible tales of the type Propp studied. In the segment visible above, for example, one can fill in the semantic holes: the villain is talking to a duckling who is the "child" who runs away, followed by the villain to a castle, which is then eaten by the villain"s serpent, with the duckling inside. Years later, the mother tells the remaining duckling about his lost sibling, and he goes off in search of the lost one. The duckling then becomes the hero, off on a journey, and meets what Propp calls a "magical agent": the leopard caught in the hunter's trap. And so on.

However, the result is a long way from the conventional story sequences one is used to. Characters jump in and out with little introduction; there is no discussion of motivation; the narrative verges on incoherence. Even when smoothed out semantically, the tale is not obviously an elegant one.

But it is the case that Propp is not interested in issues of quality in this part of his work or, indeed, in anything beyond the development of a morphology which allows classification of tales. A morphology of folk tales does not offer criteria for choosing the optimum tale any more than morphology in biology

helps one choose the most beautiful flower. There are simply flowers of one kind or another, folk tales of one kind or another.

Though the sequence of the elements is of interest to Propp, devices in the tale used to make sequences believable are not. For Propp, such devices — like characters' names, personalities, and ideas about the world — do not operate to put a tale in one sub-class or another.

Propp's observations on these matter are clear. "Motivations belong to the most inconstant and unstable elements of the tale," he observes, and, "[t]here is reason to think that motivations formulated in words are alien to the tale on the whole." Perhaps most tellingly: "One may observe in general that the feelings and intentions of the dramatis personae do not have an effect on the course of action in any instances at all." (These observations are about a particular set of Russian fairy tales; they are not about tales or fiction in general.)

When the students went back to look at the tales Propp had examined, they found that his characterization seemed to fit the actual tales as well as the generated ones. The tales, many collected from illiterate tellers, display no motivation and a lack of coherence in the conventional sense; they are full of surprises. Moreover, Propp discusses these characteristics as identifying features of the kind of tale he has examined.

Interestingly, it is possible to argue that some modern tales (Robert Coover's come to mind) exhibit some of these characteristics, too, reading like rough-hewn collections of cultural icons. That last suggestion brings the discussion full circle to Propp, who is clearly interested in looking through the jagged surface of his tales to mythic elements he believes might signal elements of a common human psychology. It is a type of interest he shares with other Russian structuralists and with the structuralist "movement" in general.

The above discussion is given bite and stimulus by students' ability to generate "results" using Propp's principles. Students discover that they cannot make a tale that has semantic coherence if they use only principles they find in Propp. His rules are all syntactic. As is the case with some Russian fairy tales, the appearance of a character requires no preparation and has no necessary relationship with anything else in the tale. Categories like Aristotle's "likelihood," "necessity," and "character" are absent from this theoretical universe, and the coherence they allow appears in computer-generated products, therefore, only by chance.

Creating a Proppian fairy tale generator with the modeling kit requires students to 1) find the key generative principles in Propp's book and translate them into formal terms the modeling kit can understand; 2) create specific examples of Propp's elements such as might appear in an actual tale; 3) determine whether they have adequately accomplished (1) and (2) by matching the resulting "tales" against their sense of the powers of Propp's analysis.

The results of the generators allow pointed, powerful, and critical class discussion of the consequences of Propp's theoretical stance and of the structuralist position in general, results we have found in other classes and when using another kind of modeling kit which we more recently devised. These experiences have convinced us that having students create their own models of a theory is a powerful pedagogical strategy. We are also convinced that technology does indeed give us the way to model theories that was not available a few years ago.

In the courses we have alluded to and others we have not discussed, we had the luxury of experimentation and support. The courses were team-taught, an expensive proposition, and students knew that they were in a "computer and literature" course and were set for the experience of holding all classes in a computer classroom. The question we now face is whether the power of this approach is to be limited to a few "computer and..." courses or whether it can be expanded to a more traditional setting.

One reason we think it can and should be is that "the traditional classroom" is changing, at least in the humanities. In the past, most departments in the humanities and social sciences followed a decades-old pattern when it came to theory. It got put in the theory course, usually taught in the junior year and sometimes called "the methods course." In English, this has long been the literary criticism course, traditionally taught as the history of western literary criticism from Plato to Northrop Frye.

More recently, teachers have begun to argue that theory is present in all courses where human behavior or products are interpreted. But it has been hidden — not because teachers do not know they have points of view and certainly not because they want to mislead students — but rather because there has not been time. One wanted to deal with the literature, to cover the material, and, at the undergraduate level, theory was thought to get in the way.

Therefore, in English, we told students that tragedy had powerful protagonists whose story went through five stages or that romance featured youthful protagonists who conquered obstacles on their way to marriage and a world of pastoral peace. We gave them quick descriptions of narrative categories or genres to help them organize "the material" as we worked through it, from "lyric" to "heroic epic."

Increasingly, teachers want to treat such categories as assumptions to be questioned and investigated, and we believe a technique based on our work offers an efficient and stimulating way to do so. As a brief experiment with volunteers in a short story course last year showed, one can bring the Linear Modeling Kit into an ordinary classroom filled with books and pencils and holding only one computer with a screen projector.

This is partly because so many campuses now have enough public access computers to allow homework assignments requiring computer use and partly because nearly all sophomores on most campuses are now at least word processing literate. Since we use mostly Macintoshes at our college, that means that students can point, click, select, copy, and type before they come to class. Since computers became widely available at Wooster about four years ago, the need to provide basic instruction in how to use them has disappeared. Moreover, some recent experiments have indicated that our program's style of syntactic modeling is relatively easy for students to grasp.

In the short story course, the question of story kinds came up in connection with a number of stories in which something horrible happened. "Are these horror stories?" asked the instructor. There were murmurs in response. "All right. Let's think about horror stories. What elements are there in such a story?"

It quickly became clear in the discussion that followed that students were most comfortable making generalizations about the kind of horror story that has appeared in recent popular films. Partly because this is a relatively uncomplicated genre, we left the more complicated stories we had been reading in order to come back to them with some firmer ideas about the value of categories.

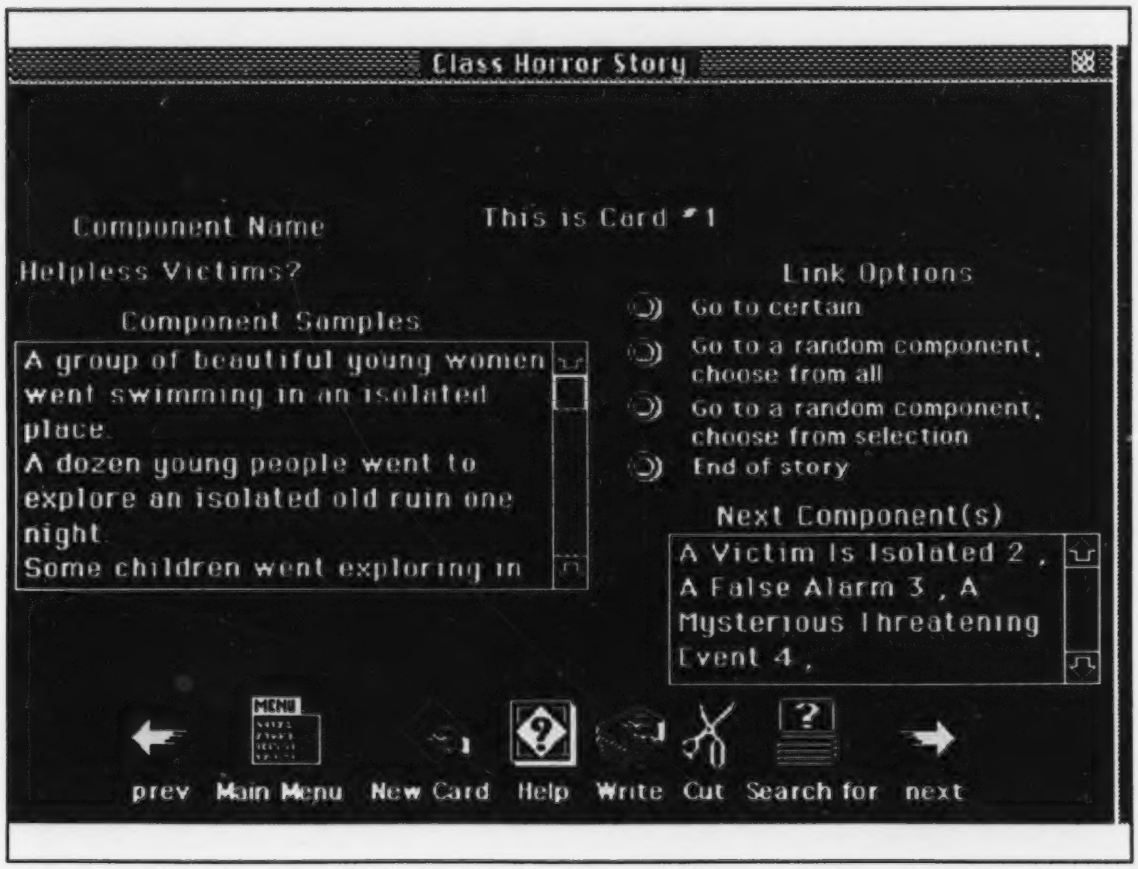
In short order, the class had produced the following list of elements they thought could be found in horror stories — in their approximate order with degrees of importance assigned.

1. Helpless victims (essential)
2. Victims isolated from help (alternative)
3. False alarm (alternative) (can be repeated)
4. Mysterious threatening event (essential) (can be repeated)
5. Warnings ignored (alternative) (can be repeated)
6. An overt, violent threatening event (essential) (can be repeated)
7. Villain revealed (essential)
8. Villain defeated (alternative)
9. Villain apparently defeated (alternative)
10. Villain promises to return (alternative) (One of the last 3 is essential)

As is often the case in our experience, the question of kinds arose here naturally from a consideration of a series of similar texts. The course we are now planning for next year will be able to take advantage of the question to lead to a unit using the linear modeling kit to sharpen the discussion of kinds. Such a discussion is of course talk about theory. But as it draws on the stories that got it started in the first place, so it improves the class mastery of the material.

After coming up with the elements that identify a horror story, the class is ready to be introduced to the Linear Modeling Kit. The introduction can be accomplished in a computer lab, if one is available, or with the help of a computer and screen projector. The rubric for the introduction is that the Modeling Kit provides a way of testing their list of elements to see whether it can produce anything that looks like a horror story.

The class is asked to bring to the introductory session at least two narrative strings for each element. That is, they are asked to present variations of (say) "helpless victims" as they might appear in a real story, such as "Some teen-agers went to explore an old ruin one night." In a computer lab, these can be typed into labeled cards by the students. Or the instructor can type them in for the class, using the screen projector:


Class Horror Story
Component Name
This is Card # 1
Helpless Victims?
Component Samples
A group of beautiful young women went swimming in an isolated place.
A dozen young people went to explore an isolated old ruin one night.
Some children went exploring in
Link Options
Go to certain
Go to a random component, choose from all
Go to a random component, choose from selection
End of story
Next Component(s)
A Victim Is Isolated 2 , A False Alarm 3 , A Mysterious Threatening Event 4 ,
MENU
prev
Main Menu
New Card
Help
Write
Cut
Search for
next

At the same time, the class can suggest the rules that should determine what can happen next, which can be defined in the Link Options. Above, there are four options once the victims have been introduced.

At this point, one can make the assignment: Create a horror story generator of your own and a generator that will produce another kind of story. (It is best to limit the generators assigned to a maximum of 10 elements, else students' ambitions quickly exceed the time available.)

We have found it best to group students into teams. Teams of three or four are almost certain to include someone who is a Macintosh adept, and, because this task requires invention, we have found that multiple heads reduce anxiety. Again, if a computer lab is available, the next session should be held there so that technical snags can be dealt with as they crop up. If there is no such lab, each student is given the program and a manual. (*HyperCard* is installed on Wooster's lab computers.)

Below are story skeletons from two such generators, the first a horror story generator already mentioned and the second a Western Melodrama generator.

Horror Story Skeleton

A dozen young people went to explore an isolated old ruin one night. Two victims sneaked away from the crowd, looking for a quiet place to talk. A victim found a pool of blood. An agonized scream proved to be only an angry cat. A very small victim, unable to walk fast, fell behind the group. A menacing shadow turned out to be just another victim. Evil laughter sounded as a victim stumbled over a pile of bones. Though smaller and weaker than the rest, the victim determined to take an independent path. A victim's pet was brutally killed. A dead victim had been strangled with a silver chain. The tattered sign read, "Go Back!," but they all laughed and went on. Then the water was smooth again. "He's gone, isn't he?" said a victim. Everyone nodded. Behind them, as they watched the sun set, the bushes moved.

(Skeleton 1 was made of these elements in this order: 1) How do I start? 2) A Victim Is Isolated. 3) A Mysterious Threatening Event. 4) A False Alarm. 5) A Victim Is Isolated. 6)A False Alarm. 7) A Mysterious Threatening Event. 8) A Victim Is Isolated. 9) A Mysterious Threatening Event. 10) The Weapon Is Shown. 11) A Warning Is Ignored. 12) The Villain Appears to Be Defeated. 13)The Villain Promises to Return.)

The Western Melodrama

You know how some times the old stories are good ones? Not this one. Frankly, Villain was short of cash and not too particular about how he rectified the situation. Heroine had long looked with favor upon Hero, but she could not bring herself to speak to him. Heroine's aunt died one day, and, to everyone's surprise, her underwear was lined with $100 bills. Heroine was her only heir. Word got around. The closer Villain came to the house, the more his horse's hoof beats sounded like the beating of a terrified heart. Closer and closer, the hoof beats sounded. Villain grabbed the money through the window and rode off while the Heroine slept. The sun set. When he recovered, Hero followed Villain — for years.

(Elements in order: 1) Start 2) Villain. 3) Heroine/Victim 4) Discovery. 5) Suspense. 6) Suspense. 7) The Crime. 8) The long chase.)

In our experience, two in-class hours are required to move undergraduate teams to the point where they can create generators like these. The actual creation takes a week or two and need not occur in class. Generators are then demonstrated and explained by each of the teams.

Before those presentations are made, student teams will have worked through the essential challenge involved in producing a generator: Which elements in which possible orders will reliably produce convincing story skeletons? They will have been adding and subtracting elements and fiddling with combinatory rules and checking output, much as a computer programmer does, as they work toward a satisfactory generator.

In the class discussions as the generators are demonstrated, one has at hand both the readings the class has done and the theoretical work required by the construction of the generators. This makes for a particularly rich environment, good for several sessions, in which an initial question is whether the generator's output adequately represents the kinds of stories found in the genre represented. Such a question naturally leads to the discussion of works read in class that might or might not fit the category in question. The result here is a more sophisticated appreciation of the nature and use of categories, often resulting, too, in a re-emphasis on what is unusual or non-standard in this story or that.

A second kind of question is whether the two generators created produce readily distinguishable kinds. Can one get something that sounds like a horror story from the Western generator? Or vice versa?

A third question, related to the first, is whether there are important elements in the genre stories the class has read that are not represented by the generators. If there are, can they be added, or is there no way to get them into syntactic form?

This discussion leads, finally, to the question of whether there are defining elements within a category that cannot be captured by a syntactic model. That this discussion happens to be at the heart of the continuing Artificial Intelligence controversy is not the point. Rather, it is a good way to stimulate talk about aspects of texts that are semantic.

Our preliminary experience in a few traditional settings shows to date that it is possible for students to use computer modeling in a literature class set in a conventional classroom and whose materials consist primarily of books. Essentially, the modeling kit gives students a concrete way of examining theoretical issues that arise naturally from their consideration of patterns and recurrent structures in the texts they have read. With the opportunity to test a few approaches to such issues by modeling their consequences, students' thinking about what they have read can become significantly more sophisticated.

Although, for the sake of explanation to a multi-disciplinary audience, we have chosen familiar, popular literary kinds as examples, our students have also used the modeling kit with literary forms more sophisticated than the horror story or the western. Our preliminary experience indicates that much useful analysis of literary structure can be formalized as the Linear Modeling Kit requires. Moreover, as students extract what they can formalize from their theoretical inferences and process it with the Linear Modeling Kit, they improve their feeling for the consequences of theoretical assumptions and their understanding of the texts theory aims at characterizing.

What we have had less opportunity to test is our conviction that the modeling kit can be used to model theories in most areas of the humanities and social sciences. Clearly, the application is not limited to generating skeletons of literary narrative. It should be useful in helping students think about any syntactic theory, whether in psychology, sociology, or political science.

Selected Bibliography

Eagleton, Terry. Literary Theory: An Introduction. Minneapolis: University of Minnesota Press, 1983.

Graff, G. *Professing Literature: An Institutional History*. Chicago: The University of Chicago Press, 1987.

Havholm, P. and L. Stewart. "Modeling the Operation of Theory." *Academic Computing*, 4(6), 1990.

Landow, George P. *Hypertext: The Convergence of Contemporary Critical Theory and Technology*. Baltimore and London: Johns Hopkins Press, 1992.

Propp, V. *Morphology of the Folktale*. Trans Laurence Scott. 2nd ed. rev. and ed. Louis A Wagner. Austin: U of Texas Press, 1968.

Sharples, M., Hogg, D., Hutchison, C., Torrance, S. and Young, D. *Computers and Thought: A Practical Introduction to Artificial Intelligence*. Cambridge: MIT Press, 1989.

Stewart, L. and Stevens, B. *A Guide to Literary Criticism and Research*. New York: Holt, Rinehart & Winston, 1987 (2nd revised ed., 1992).

Telford, Kenneth. *Aristotle's Poetics: Translation and Analysis*. Chicago: Henry Regnery, 1967.

Multimedia in Education

Carl W. Helms
Doris R. Helms
Educational Information Technology Laboratory
Clemson University
Clemson, SC 29634-1910
cwhelms@clemson.clemson.edu

In the sciences, in particular, class sizes have been increasing in recent years. In addition, with the growth of scientific information and the need to integrate key concepts across fields, our ability to grasp information has become increasingly visual. Recognizing that chalk and blackboards and transparencies with markers are not sufficient to present this information to undergraduates in an understandable manner, the College of Sciences at Clemson University established the Educational Information Technology Laboratory (EITL). EITL has two major purposes: (1) the enhancement of multimedia software designed for interactive presentations (software development) and (2) assisting faculty with instructional design, capturing multimedia objects (visual, text, and sound data), and building applications for use in lectures or individualized instruction (faculty support). Working with the Center for Excellence in Science and Mathematics, EITL also supports media development for K-16, the dissemination of educational technology throughout the state, and pre- and inservice teacher training--particularly the training of those who teach teachers and of lead-teachers/media specialists in the districts.

Why multimedia?

How does this technology differ from that currently found in many schools? "Multimedia" describes the computer controlled access to multiple media--visuals (including full-motion video), text, and sound--in an *interactive* environment. Why use a computer? Instructors can combine slides, prepared transparencies, and video sources in the traditional classroom. These techniques do, in fact, improve traditional instruction. However, the computer provides integration of media resources and assists the teacher in making the resources visible (proper size, color, legible fonts, etc.) Moreover, it is far easier to capture a visual in digital form on the computer than it is to make a slide. It is also easy to edit resources. Thus the technology is more likely to be used and to be used correctly. It is also interactive and under the control of the lecturer. The availability of multimedia objects and the work done in arranging them for lecture also provides resources for authoring stand-alone training modules for use in the learning laboratory or in free-standing kiosks. Digital applications can also be networked, making resources available to a larger group outside the traditional classroom and the real-time lecture. Review, distance learning, access and equity, and the non-traditional learner are all served. Learning can also be assessed directly using multimedia systems.

Multimedia Systems.

At present, a number of multimedia technologies are becoming available--systems and software are evolving rapidly. Our multimedia operating software was developed by AT&T Bell Labs' interactive TV (ITV) group formed in 1981. It forms the basis for WorldKey System at Walt Disney World Epcot Center, the 75-screen video wall at the Bedminster Network Operations Center of AT&T, and recent installations of interactive games at Universal Studios in California and Florida. Several individuals from the ITV group came to Clemson in 1989 and AT&T awarded Clemson the exclusive license to continue development of the software system. Based upon the original DOS-based operating kernel and scripting language, the group has developed both graphical authoring ("LighTable") and presentation ("SoapBox") applications. At the heart of this system is Truevision's Targa graphics adapter, a card that accepts video and input and provides video as output. The use of NTSC (or PAL) video provides a stable standard while other graphics technologies change monthly. The system also offers direct support for distance learning and educational television, offering a video source that can be integrated into cable or broadcast feeds directly. A sound card provides capture and playback capability for digital (waveform)

audio and a multi-port serial adapter allows the computer to control multiple external devices (mouse, touch screen, videodisc player, scanner, etc.)

Authoring.

In assembling materials for preparing lectures, instructors typically gather their visual, textual, and auditory resources. Still visuals can be captured as digital files using a flat-bed scanner, a slide scanner, or a video camera, still or motion (video sources are digitized as graphics files). These files can be edited, sized, combined, etc. using any graphics or paint package and saved in any of a number of graphics formats. Text can be added as an overlay so it can be changed or edited without affecting the background. Sound sources (waveform audio) can be digitized using a microphone, tape deck, or other source connected to a sound card in the computer. Live video can be incorporated as it comes from a camera, video tape, or a video disc player. Full-screen, full-motion video remains an analog technology in most multimedia applications at this time.

Video? More about motion--various methods can be used to portray time- dependent events digitally. Animatics or animation can provide the appearance of motion. Various authoring packages support the creation of simple animations based upon the playback of separate digital files, usually at 8-15 frames/sec in a window that is less than full-screen, giving the appearance of motion,. QuickTime extensions to system 7 for Macintosh computers provide scalable motion capabilities that are often sufficient to convey all necessary information to explain a concept or illustrate a process. In the IBM world, Digital Video Interactive (DVI) offers low quality, full-screen, full-motion digital video. With increased processor speeds, the development of graphic coprocessors with separate video memory, improved bus bandwidths, and improved compression algorithms, symmetrical processing of digital video in real time will be possible and will become integrated with the world of digital HDTV. For now, however, economics favor the integration of digital and analog sources in providing multimedia for instructional use. This can be done seamlessly.

Once instructional materials are organized and captured for use in class, they are organized into a linear sequence using an authoring system. Text overlays and sound are added at this stage. A "show" is then compiled--this adds common touch-points and other features that can be accessed during class. This "show" is then saved to the hard disk or to removable media (high-density floppy, Bernoulli, Magneto-Optical, Write-Once-Read-Many disks, tape, etc.). Removable media can be used to transport the lecture materials to the classroom (they could also be downloaded by network connections or over high-speed telecommunication links). In our multimedia lecture facilities, we have an 80386 microcomputer with the required adapter cards attached to a video projector and sound system. A videodisc player is also available. Pointing devices include a touch screens and mouse. The touch screen is favored in the classroom--reaching for a mouse and registering precise locations on the screen are more difficult than touching the monitor face directly.

The Lecture.

Images from a lecture called "Circulation" will be used to illustrate the use of materials in class. The instructor can move forward or backward, one graphic at a time. In response to questions or to refer to an object out of sequence, the instructor can call up "menu" screens containing the entire lecture in groups of 16 images/screen. In addition, any image can be taken into an "Electronic Blackboard" where, using a touch screen or mouse, lines and text can be added or graphics superimposed. The original image can be cleared and a new screen can be built using "clip art" images or the mouse and keyboard. Animation sequences will also be used to illustrate instruction involving time-dependent events.

Non-linear presentations (a lecture on animal diversity, called "Zoo," and one showing cellular organization, called "Cell Tour") will be used to show the added interactivity permitted to the instructor using computer-based systems. The use of information from these lectures to develop stand-alone instruction modules will be illustrated. Interactive applications can be distributed using compact disk interactive (CD-I) if this technology takes hold.

The uses of multimedia technology described thus far require only one system for a group of students--not the one-on-one machine/student found in the typical learning laboratory. Much of the interaction is guided by the instructor but reaction to questions or incorporation of new material is possible. Student response units can also be used to provide limited student interaction to questions in the classroom. It is also possible to allow individuals or groups to use multimedia systems to create lessons/reports based upon resources provided by students, their instructors, or accessed from central media resource libraries. These projects can be quite innovative and may represent major syntheses of information and concepts, often crossing the borders of traditional disciplines. They can form portfolios used for assessment of individuals and programs that are tied to new criteria.

The Good and the Bad.

Advantages to our Targa-based technology include (1) a robust software based on a concurrent operating system, allowing complete user control of the presentation, (2) the ability to incorporate any video source and to provide video output that can be recorded, fed locally, or broadcast directly, and (3) a color palette of at least 32,000 colors (as opposed to the 256 limits of normal VGA). Disadvantages include incompatibility with VGA and other display standards.

Nuts and Bolts.

How does all of this work in the real world with limited resources? At Clemson, the College of Sciences made the commitment to get EITL off the ground. We, in turn, have formed a service arrangement with a commercial group (Advanced Interactive Multimedia, Inc., of Raleigh, NC) to market services of the group. It is our hope that funds derived from commercial activities will allow us to continue software development and faculty support.

How do we provide support to many across campus with a small staff? Use of this technology has grown as faculty become aware of its potentials and as large classrooms are equipped to support multimedia presentations. Our initial efforts in biology have produced a group of faculty members able to handle 95% or more of their own authoring--they have a creation station and image capture hardware. They return to us for advice on instructional design, animations, videodisc production, etc. However, they have become essentially self-supporting. Chemistry and the modern foreign languages are currently working on the earlier slope of this curve. Others will follow as these groups are trained and weaned. We hope to be able to provide portable systems that will allow instructors to use multimedia presentations in smaller classrooms not permanently equipped with computers and video projectors.

While developing object libraries (media resources) for use at Clemson, we recognize the need for multimedia resource libraries available throughout the state and for use in the K-12 world. This will require support groups such as EITL at other universities. These groups can provide equipment, software, and personnel resources for creation, indexing, and archiving of materials, adding to the object libraries. The state's network of Comprehensive and Technical Colleges can provide technical support and training locally to faculty and inservice teachers. As workshops are conducted, teachers will bring a wealth of resources from their own experiences (slides, tapes, etc.) to use in their lessons. These resources can be added to developing resource libraries. ETV has miles of videotape available for use in the classroom. One of the major tasks ahead of us is to index these resources so they can be retrieved and used.

Note that multimedia objects must be "device-independent." They must be usable on IBM machines with programs ranging from LinkWay (IBM) to Multimedia Toolbook (Asymetrix) and Authorware (Macromedia) and on Macintosh computers using Hypercard (Claris), MacroMind Director (Macromedia) or Authorware, etc. Standards will develop, but in the current graphics babel, video standards remains usable on all platforms.

As libraries are developed, how can they be accessed? Catalogues can be available on-line (in South Carolina, a Wide Area Network, CUFAN, is accessible to all teachers and SC ETV is installing Learning Link with national (Internet) connections. However, the movement of multimedia materials themselves will require

"sneaker net" or mail, advanced communications technologies, high-speed networks, or data feeds via satellite. All of these methods are possible and are being explored.

Again, Why Multimedia?

1. ***Multimedia reaches all of the senses***--sight, sound, touch. Students remember 20% of what they see, 40% of what they see and hear, and 75% of what they see, hear, and do.
2. ***Multimedia can be tailored to all learning styles and levels***--visual, verbal, auditory, and physical. It serves traditional and non-traditional students and on-site and distant learners. It provides for remediation and amplification.
3. ***Multimedia encourages self-expression and pride of ownership.*** Teachers are not locked into others' instructional designs and can fit presentations to student levels and the learning situation.
4. ***Multimedia fosters cooperative learning and communication.*** Students learn together and discuss their perceptions and understanding.
5. ***Multimedia creates an active learning environment.*** "Hands-on, minds-on" makes students think. The multimedia computer does nothing without interaction.
6. ***Multimedia is patient, offering a consistent level of instruction.*** Lessons can be repeated until learning criteria are met. Gender and ethnic (and language) barriers are reduced.
7. ***Multimedia is FUN!***

Will multimedia change the learning environment? Yes, it already is. Can teachers change from being the source of all knowledge to being mentors--facilitators of learning? Yes, those we have talked to are excited about the creative opportunities in education. Technology, used by dedicated and understanding faculty, has the potential to revolutionize learning and to help establish a pattern of life-long learning for the citizen of the world of tomorrow!

Cross-Curricular Computing in a Liberal Arts College

James A Koger
Academic Computing
Lynchburg College
Lynchburg, VA 24501

The passage of a faculty resolution on December 1988 supporting an institutional program to get computers into the hands of faculty and students marked the beginning of Lynchburg College's Knowledge Initiative. There has probably been at least as much debate over the label as over most of the details of the program, but the essential purpose of that resolution was to make the computer, or more accurately all the resources of information technology, an essential tool in teaching and learning throughout the curriculum.

Implementation planning was done by a Knowledge Initiative Task Force, formed in February, which consisted of faculty, administrators, and a few Trustees. During the summer, the Task Force evolved into a Knowledge Initiative Committee, with roughly the same composition minus the Trustees. A new full-time Director of Academic Computing was hired in the summer of 1989 after a nationally advertised search, and 1989-90 became the year for faculty implementation and training, while students were given an option to purchase a computer through the college. Zenith Data Systems was selected to supply the computers.

During the summer and fall of 1989, a vigorous discussion ensued about the inclusion of Apple as one of the choices for students for the fall of 1990, the first full implementation. Some argued that supporting a dual platform on a small campus was not necessary or feasible, especially when there was already a large installed base of DOS machines (about 150); others, primarily theater and art department faculty, said that their majors needed to have Macs, and that some choice would be good for all incoming students. In the midst of this discussion was injected a complicating factor, an administrative initiative for a graphics lab given by Apple. The final result was that without a genuine resolution of the issues, the administration declared that "Apple would be a partner." Therefore, during the first two years of the KI implementation, a DOS (IBM-compatible) and an Apple computer were offered to the students through the college. The predictable support problems did indeed occur, and after the first two years, one of the changes was the dropping of the Apple purchase option.

Beginning in fall of 1990, full-time traditional students were required to purchase or bring a computer which met the standards established by a platform sub-committee of the KI committee. The exclusion of non-traditional students from the requirement was based on the assumption that many of these students would not be able to afford the added expense, and that for the relatively small number of non-traditional students at LC, public access labs would provide sufficient access. One purpose of the requirement was to allow financial aid to help support the computer purchase for needy students; another was to make sure that curriculum planning could proceed with the expectation that all the students in that entering class would have access to a computer for course work.

The platform sub-committee had initially been leaning toward a laptop, similar to the solution chosen at Drew University, which was one of the influential models for Lynchburg College's program. But as we examined what we wanted our students to do, we decided that some of the software, especially in business and the sciences, really worked better on something more than an 8088 processor, and prices for 286 laptops were simply too high. Portability vs. processing power was decided on the basis of a better processor, especially in what we hoped would be a four-year machine.

Another difficult decision was the choice of a software package. The Board of Trustees had imposed a ceiling of $2100 for the system, including printer and software, and even though we were a WordPerfect campus, $160 for each student copy of WordPerfect put the system over the top. After a careful examination of several solutions, Microsoft's Works 2.0 was chosen for its combination of low cost, integrated function, and ease of use. DOS 4.0 and Windows 3.0 were also included. There was some debate on whether Windows should

be included, but those who felt that the graphics interface, especially for primarily novice users, was important and would become more important in the future, won the decision.

So the platform for the 1990 entering class was a desktop 286 with mouse, VGA color monitor, DOS, Windows, and Works, and a dot matrix printer. Zenith Data Systems again was selected as the DOS vendor for 1990, and offered the LP/8, a 286 computer with a 20 MB hard drive and one 3.5" floppy, a VGA color monitor, and an Epson LX-810 dot matrix printer. The Macintosh SE with a monochrome monitor was the choice on the Apple side. There was some question about why the Mac was not required to have a color monitor like the Zenith, but the additional cost of that option for the Mac was allowed to stand as reasonable grounds for the difference. Orders were collected from students through mid-summer, and at the end of August, distribution of over 350 computers was handled in a three-day period. Actually, distribution of only the DOS machines happened then, because the Apple order was delayed for a couple weeks, and when the Macs did arrive, some did not have the correct memory, and another problem had to be overcome for the patient Mac users.

Many of the courses that were predominantly freshman had been working on instructional software, so some of the freshmen were able to use courseware, besides meeting some of their assignments by more conventional use of word processor, spreadsheet and database features of Works. One example of courseware is FineLines, a package that accompanied the Beacon Handbook, used in Freshman English. It featured an on-line handbook, several aids to invention(ways to begin a writing assignment), some interesting editing components, and a grading utility for the teacher which provided 20 macros for common writing mistakes and some nice tools for commenting on a student's "paper" while still on disk. Some faculty outside English who also read a lot of essays found this package attractive enough to use in their courses, so it saw use outside Freshman English.

The Macintosh's were ordered with Soft PC, software which was supposed to allow the Mac user to run DOS-based programs; unfortunately, the level of difficulty for the user meant that many of the 30 freshmen who had chosen Macs simply gave up on using packages, such as FineLines, which did not come in a Mac version. Faculty who were accepting work on disk rather than in hard copy ended up getting student work done almost exclusively on a DOS machine, rather than on the owner's Mac.

Faculty training on the software, including Works, Windows, and Toolbook (which was included by Zenith on all faculty and student machines as an inducement for adoption), was provided throughout the summer and fall of 1990, and training on the use of the one networked classroom, which had actually predated the official beginning of the Knowledge Initiative (installed summer, 1988), encouraged faculty from many disciplines to take advantage of this instructional facility. A variety of application software was featured in this classroom, which used a 386 server, Novell 2.x Netware on ArcNet, and 26 8088 work stations with local hard drives and EGA color monitors. The principal use of this classroom, however, was to facilitate class discussion over the Daedalus Interchange. Lynchburg College was the beta test site for the Daedalus system during the 1988-89 academic year, and classes in English, history, business, political science, nursing, physical education, and computer science vied for time on the 4th floor of Hopwood Hall. The popularity of this classroom and its capabilities led to the creation of a second-generation version--386 work stations, recessed VGA monitors under a glass panel, Ethernet, and an upgrade to Novell 3.11 in the fall of 1990.

Interest in the instructional use of Toolbook was piqued by the training, and two relatively novice computer users in political science, Joe Freeman and Dan Lang, began working on a program on the U.S. Constitution they called "TextMaster." With the considerable aid of a sophomore computer science major, Fred Price, who had taken an early interest in "Toolbook, they came up with an instructional package which was selected as one of the Joe Wyatt "success stories" for 1991. Other faculty, encouraged by their colleagues' success, have also begun other examples of Toolbook applications.

1990 saw the departure of the first Director of Academic Computing because of irreconcilable differences with the administration, and an acting director of campus computing--the current administrative computing director--was appointed to take charge of the Knowledge Initiative. Since this person's position at Lynchburg College was restricted to administrative, VAX-based computing, however, and because the faculty

were beginning to feel out of touch with the program, this administrative solution survived only part way through the next year. Late in spring of 1991, after a consultant's visit, a new Director of Academic Computing came out of the faculty ranks to try to manage the Knowledge Initiative.

The DOS vendor choice for 1991-92, made early in spring 1991, was IBM, and a new Mac, the LC, became the Apple product for Lynchburg students. Price drops in the market during the late spring and summer of 1991 produced little change in the IBM costs, however, and students ended up paying their $2200 for a 286 with a 30MB hard drive, color monitor, printer and the same software as in the previous year. Apple adjusted its prices a little more, although the Mac LC was, of course, more expensive than the IBM Model 30. Due to the changes in the economy, Lynchburg, like many other schools, saw a dramatic increase in the level of need for financial aid, and not just for the new class. Cost overruns in financial aid came to almost a seven-figure total, and part of this overrun was the full purchase of 131 systems which were given to needy students. The ingenious solution for individual student ownership of a computer through the help of financial aid had just backfired.

1991 saw more changes than the directorship in academic computing. Faculty concern had seen the replacement of the ad hoc Knowledge Initiative Committee with the regular faculty Academic Computer Services Committee as the principal advisory group for the Director of Academic Computing and oversight committee for the Knowledge Initiative as a whole. Recent IBM pricing policies prompted an examination of an early assumption, mostly Trustee-based, that parents and students needed a "name-brand" computer to give the program credibility. Initial costs and to an even larger degree, maintenance, made the college look closer at the possibility of moving to a generic platform--a position that had support from some faculty early in the KI planning. The financial aid fiasco gave the final impetus to this position, and, rather surprisingly to some observers, the administration approved a change in policy to include generic computers in its consideration of hardware. Complications in hardware support for college-owned equipment: replacement parts unavailable or delayed because of single-source supply, compatibility problems with Micro-channel hardware and some of the software that was intended for these units, and difficulties in upgrades also influenced the change in policy.

Even lowering the cost of the student unit by going to a generic computer did not entirely solve the budget problem, however, and after a lengthy discussion throughout the campus community, the accepted solution was to use much of the money set aside to support the Knowledge Initiative through financial aid to provide a base under all the student computers for the next year. Each full-time entering student would be charged a "user fee" of $500, and after that fee, they would upon graduation from the college, own the system. Those who left before graduation could take the computer/monitor/printer with them if they paid the difference between the user fee and the system cost to the college. If they don't want to take the computer, the college gets another machine to recirculate. Out of the 150 incoming freshmen who have responded so far, 6 have elected not to participate (5 have indicated that they are bringing their own system). Yes, this time there is a choice. To take the whole thing out of financial aid, the individual student is not required but strongly recommended to bring or purchase a computer. Some faculty have feared that this shift in policy would leave a great many students without a computer, but $500 for a 20MHz 386SX with a 40 MB hard drive, 4MB RAM, high density 3.5" floppy, VGA color monitor (.31 dot pitch), Epson printer, DOS 5.0, Windows 3.1, and WordPerfect for Windows 5.1 has been too good a deal for most prospective students/parents to turn down. One of the benefits of the college retaining ownership is that we can use site license software at considerable savings. It's too early to say what kind of impact this offer has had on enrollment or on participation in the program, but response from parents and students so far has been uniformly enthusiastic--even incredulous ("Where's the catch?" "Is there fine print somewhere?" "$500 only once?!"), and fees paid applications are running about 20% ahead of last year's. We are not out of the woods yet, but it feels, for the first time, as if we may be heading in the right direction.

The idea of a full campus network had been part of the goal of the Knowledge Initiative from the beginning, but several network planning committees had been formed and then discouraged by a lack of administrative support even for planning. This spring, however, a Network Task Force, backed by the ACSC and accepted by the President, has begun work on a comprehensive network plan, to be finished this summer.

This includes academic and administrative computing, faculty and administrators, and students. We think that the future of a campus network is now beyond just a virtual one.

Some reflections on aspects of this brief overview of a 4-year process are in order. The lack of clarity in delineating responsibility for program management at the outset was ultimately responsible for many of the subsequent problems. It is absolutely imperative that the administration and faculty agree on a process for determining such matters as implementation, budget planning and management, hardware and software selection, and support and training issues. We have found that it is almost impossible not to underestimate the amount of training that will be needed, especially if the implementation includes, as it does for a liberal arts campus, a wide range of disciplines, amount of experience with computers and, even more significantly, with a hodge-podge of software. We have standardized on much of our common software: word processing, spreadsheet, database (sort of), and a statistical package, but that still leaves God's plenty in disciplinary and instructional software.

We have found that exposing faculty to ideas in using computers in the teaching and learning process is important. We have paid expenses for faculty going to EDUCOM, and we have tried to send different faculty so that more can get that experience. We have sent people to the IAT in Research Triangle Park, NC, and we have brought technology leaders in the various disciplines to do demonstrations. We have software demos, software training, we share tips on how to use some of our standard packages in the classroom--in short, we try to get as many faculty involved in using the computer and in having their students use the computer as possible. When we get our Internet connection next month, I'm sure that will open up a whole new world for our faculty and for us as we begin to try and support one more application.

"This Stuff's Not Contagious!" Why? And What Can We Do About It?

Joseph McDonald
Lynda Micikas
Pepperdine University
24255 Pacific Coast Highway
Malibu, CA 90265

Approximately three years ago, Joe Wyatt, Chancellor of Vanderbilt University, suggested to the academics attending an EDUCOM conference that they locate 100 examples of successful uses of information technology (IT) in higher education and publicize them. Commenting that "this stuff's not contagious", Wyatt observed that applications of information technology in specific courses or for specific educational experiences never seemed to be used by anyone but the people involved in their creation and a few colleagues. "I would like to see someone develop a list of 100 uses. We ought to see if we can get them implemented in our institutions" ("Vanderbilt's Chancellor: a Tireless Advocate for Computer Technology", 1991).

EDUCOM decided to take Mr. Wyatt up on his proposal, and even to go one better, in what it called "The Joe Wyatt Challenge". As Judith Boettcher, chairwoman of the project and assistant director of Education Technology Services, Pennsylvania State University, noted, "Our basic goal was to answer the question, Is information technology making a difference in higher education?" EDUCOM's assumption was that computers in the classroom can improve teaching and learning and it set about finding the evidence, under the auspices of one of its programs, Educational Uses of Information Technology (EUIT).

Joe Wyatt entered into the search with enthusiasm. He sent a letter to the president of every college and university in the United States and Canada asking for examples of the successful use of IT in their institutions and received 288 answers. These classroom applications of technology were studied by a blue ribbon committee of 28 educators, who selected 101 examples of responses to the Wyatt Challenge which were then announced in the EDUCOM national meeting, October, 1991. The list and description of the winning projects and the faculty members associated with them was published in the Chronicle of Higher Education, October, 1991.

It is important to note that the Wyatt Challenge and Chancellor Wyatt's observation, "This stuff's not contagious", although closely related, focus on two different phenomena. The Wyatt challenge sought to find examples of computer technology used successfully in specific classrooms or learning situations. Undoubtedly, it succeeded. The list of 101 applications contains what appear to be some effective uses of computing in teaching and learning. But Wyatt's complaint addresses another issue, namely, that even the successful uses of IT do not seem to spread, but remain useful only in the courses or experiences for which they were specifically designed. The answer to the question, "Is information technology making a difference in higher education?", cannot, on the basis of the Wyatt Challenge, be an unqualified, yes. It might be making a difference in 101 or 288 specific instances, perhaps. But if IT remains non-contagious, how much of a difference is it really making? And why is it not contagious?

We propose that IT is not contagious in the classroom principally because teaching and learning are so thoroughly individual actions that specific application of technology in any specific act of teaching or learning almost precludes it from successful application in some other specific instance of teaching and learning. Furthermore, we suggest, in teaching and learning the fundamental issue is not technology, computer or otherwise. To attempt to insert technology into the learning process, simply because technology can be used to accomplish certain tasks related to learning is approaching the matter from the very worst possible direction. It is not technology that needs to move into the curriculum. Rather, it is the curriculum and learning strategies and pedagogy that need to appropriate technology, to move into technology, as it were, as this is found useful and desirable for learning. Finally, we maintain, the relationship between teaching and learning and IT is too narrowly conceived. Information technology is considerably broader than computers. The more basic and

important question is the relationship between learning and the total information environment in which it is embedded. Technology is only one piece of that environment.

TRENDS IN EDUCATION

At a recent conference (Information Technologies: Critical Choices for Challenging Times, Pittsburgh, April, 1992), Robert B. Kozma, associate professor in the School of Education and, director, Center for Research on Learning and Teaching, University of Michigan, discussed six significant trends in teaching and learning. In his view, IT is a significant factor in making these trends possible, in moving them along. These six trends or developments are:

1. From Reception to Engagement

   Students are moving away from passive reception of information to active engagement in the construction of knowledge.

2. From Isolation to Interconnection

   We are moving from a view of learning as an individual act done in isolation toward learning as a collaborative activity. We are moving from a consideration of ideas in isolation to an examination of their meaning in the context of other ideas.

3. From the Classroom to the Real World

   Considerable effort is being made to help students break down the walls between the classroom and "real world" experiences, situations, and applications.

4. From Coverage to Mastery

   Increasingly, the student is provided with a range and variety of experiences which encourage him to "internalize", to master the material to be learned, rather than merely being exposed to a defined amount of content.

5. From Mechanics to Understanding in the laboratory

   The mere mastery of certain mechanical laboratory skills and procedures is no longer seen as sufficient. Rather, students are expected to understand and "contextualize" their laboratory experiences.

6. From Products to Processes

   Both teachers and students are moving past a restricted concern with the products of academic work to the processes used to create knowledge and solve problems.

In Kozma's view, with information technology we can accomplish the goals implicit in these trends. He declared, however, that we have not yet arrived at the place where we can or are using IT in a significant way to reach these pedagogical objectives. His statement not only echoes Wyatt's complaint, but reflects a recent question by the president of West Chester (PA) University who asked his associate vice president for information services, "Why, after several years of significant investments in technology, so little had changed in the classroom" (Staman, 1990, p. 34).

Many of those who have recognized the problem of the limited spread of IT into college courses have also suggested solutions. But these solutions almost always involve some strategy for "integrating" technology into teaching and learning, for "getting IT used", without, perhaps, a full understanding of why it has been so slow to be adopted in the first place, and why, until this question is answered, real integration might be impossible to achieve. The solution, it seems, for IT's limited popularity is to find ways to make it more popular (e.g., to raise faculty awareness of its potential, to remove the obvious practical obstacles to its use). The solution has not been, it seems, to ask what it is about teaching and learning that makes integration so difficult even after these obstacles are removed, and then to shape a response based on the answer.

At West Chester, for example, the response to the question why had IT, purchased at some considerable cost, not made a difference in the classroom, was to develop a three year plan for "infusing technology into the teaching/learning process". As we discuss below, although the plan is described as being based on an understanding of why technology had not been more successful on that campus, and although the author makes a number of very important comments about the relationship between the adoption of IT and the structure of a college course, close reading suggests that the author was, nonetheless, driven by the technological imperative: technology had been purchased and now those purchases had to be justified. IT could do certain good things, and now the goal was to get it used, on the acknowledged, but, in our minds, still questionable assumption that issues of "whether" and "why" had already been sufficiently addressed, and the only issue remaining was "how".

Likewise, Kozma, after acknowledging IT's failure to revolutionize education, identified three challenges to its increased influence: (1) "the need to move from isolated experiments to organizational commitment" (i.e., it is not enough that it is successful here and there -- it must permeate the whole institution); (2) "the need to move from pockets of innovation to models of implementation" (i.e., these pockets of innovation are good and must be expanded), and (3) "the need to move from blind faith to systematic evaluation" (but, one wonders, shouldn't this come before the first two?). He listed a number of items which would be required to accomplish this, including increased funding, shared success stories, contextualized evaluation, and, most importantly, full "integration into the organization".

And, finally, the original intent behind the Joe Wyatt Challenge was first to find successful applications of IT and then to publicize them in an effort to get them implemented in similar institutions and courses across higher education. Again, the working assumption is that more IT is good, and that if IT has been found to be helpful in one case, it will necessarily be found helpful in another. If what was going on in the other situations did not include IT, then it probably wasn't very good, and needed the noblesse oblige of someone else's IT.

We agree that the ideal condition is one in which all of the benefits of technology are fully and appropriately available to all of those involved in the teaching/learning process. But we suggest that the attempt to bring about the integration of IT with teaching and learning, simply because it "can" do certain kinds of things, and without a proper and full understanding of why it is not contagious in the first place, is, at best, foolish, and, at worst, detrimental to the long-term expansion of its educational use. Kozma's description of current trends in curriculum and pedagogy is fair and accurate. But the advances in teaching and learning which he describes are not the exclusive property of IT. Truly good teachers have been "engaging" students in the active construction of knowledge for centuries; truly effective group activities (which can be designed completely without IT) have been helping students learn collaboratively for generations; and truly good laboratory experiences have been providing students with opportunities to experience and contextualize "real" reality for decades.

Nor are these trends likely to come about more quickly simply because of the inclusion of IT in a particular course or set of courses. It is no doubt true that "good" IT can sometimes help teachers do their jobs more effectively. But it is important to remember that the power of information technology is in the pedagogy, in the "good" teaching that created the programing, and in the "good" teaching that knows how to use the programming effectively within a specific and full learning situation. The power does not lie in the technology itself. Even the best in technology cannot make up for poor teaching.

It may be true that higher education is becoming more sensitive to the aspects of teaching and learning that Kozma has outlined, and that IT can help in significant ways. But, again, we raise and expand on the question Wyatt raised: why, after 35 years of computer uses in education, have we come no further than we have in finding a place for IT in teaching and learning? And our answer, which we discuss below, is that the computer based information technology community, just like the print based information community before it (i.e., librarians), has proposed answers to questions and problems which teachers and students did not know they had. The blanket insistence on "integrating" IT with the curriculum will not work any more than have the efforts to "integrate" the library with the curriculum. What is needed, in both cases, is just the reverse: teaching and learning need to bring information (defined broadly, not just as IT) into teaching and learning, where it is needed, where it belongs, and where it will help to create a harmonious and balanced learning environment for an enormous variety of student learning and faculty teaching styles and objectives.

WHY ISN'T IT CONTAGIOUS?

In a recent paper (Staman, 1990), E. Michael Staman describes a plan to infuse technology into teaching and learning at West Chester (PA) University. Staman is an experienced teacher, administrator, and information technology specialist and makes some very perceptive observations about IT's lack of popularity. The West Chester plan is thoughtful and contains good insights into the relationships between IT and teaching and learning. But the plan, also, can serve as a case study on the central reason IT is not contagious. Staman notes (p.35 ff.):

> Given the number of successes reported in recent years it would seem that by now the use of technology in teaching/learning environments would be as common as the use of other resources available to faculty, or that we would at least see momentum in that direction sufficient to convince us that the use of such resources would become commonplace during the next few years. But the use of technology in pedagogic environments is not commonplace, and what momentum exists is developing at an excruciating slow rate.

Staman goes on to list some major obstacles to the development of that momentum. These include a faculty's lack of awareness of IT's potential and of successful uses of IT across disciplines. He also observes, "The problem is further exacerbated by more mundane things such as a lack of detailed technological expertise on the part of most faculty, insufficient staff support, lack of resources, minimal to no administrative support or commitment, a general lack of focus on the problem" (p.36).

But, in our view, Staman's most astute observation relates to the nature of teaching and learning.

> More fundamental than these kinds of obstacles, however, is the question of what truly happens when a member of the faculty walks in front of a class and begins to teach. It (the act of teaching) is a very special event, highly individualized, unique to a given professor in a given environment, teaching a given lecture in a given course. The issues are curriculum restructuring and courseware portability (in the pedagogic, not the technical sense) because the way in which a particular course is actually taught depends upon a specific professor at a specific university and is typically a function of the specific tools available.

We believe that Staman has captured the heart of the issue in locating the core of the "lack of contagion" problem in the teaching/ learning environment. Yes, the widespread use of IT is inhibited by such practical problems as cost, lack of technological expertise, insufficient staff support, and a faculty's lack of awareness about the potential of IT to support their teaching efforts. But we would argue that even if these obstacles were overcome, IT would still not be widely adopted on our campuses and in our classrooms. Teaching is, as Staman states, a highly individualized act. In fact, the structures of most individual courses probably depend more on the attitudes, interests, and personalities of the professors involved than on any external factor, such as curricular position, departmental requirements, or, even, and here we disagree with Staman, the specific tools available.

BEST COPY AVAILABLE

Individual college courses are "put together" by individuals, individuals who have particular visions for the learning which is to take place, individuals who have different comfort levels with various pedagogies, individuals who were taught by other individuals, and who tend, often, to teach as they were taught. The relative underuse of audio/visual tools, for example, or of teaching techniques other than the ubiquitous lecture, suggests that even the "availability" of other teaching resources or techniques makes little difference in the internal design and execution of most college courses.

But Staman does not follow through to develop his argument completely. If it is true that teaching is, as he says, "a very special event, highly individualized, unique to a given professor in a given environment", then it is not surprising that preprogrammed software, even very good preprogrammed software, is not contagious. What has been developed by one professor to fit his unique teaching style, objectives, and learners is unlikely to fit well or easily into another course, designed as it was by another individual to fit his style, his objectives, and his learners. In fact, it is likely that an inverse relationship exists between usefulness in one course and ease of transfer to a second: the more successful the software is in the course for which it was designed (i.e., the better the fit between the overall structure of that course and the software), the less easily it will be exported to another.

Nor does Staman seem to understand the special relationship that "external" resources like textbooks, audio/visual materials, libraries, and information resources have to the structure of most college courses. He correctly points out that most faculty are users, not developers of teaching/learning materials. He implies, however, that the use of textbooks is functionally and conceptually equivalent to the use of these other resources: "In the case of written materials, the use of resources prepared by others as tools for instruction has been occurring since the beginning of time . . . given the number of successes reported in recent years it would seem by now the use of technology in teaching/learning environments would be as common as the use of other resources available to faculty . . . "

But this equivalence simply does not exist, at least not in functional terms. Faculty tend to use textbooks largely as "support" items, separate from the day-to-day classroom process. Yes, they may design the sequence of topics to match the textbook selected. But most faculty see the textbook as a student-focused resource, one which will provide information, explanation, and sometimes question/
problem-based drill and reinforcement, but will not play an important role in the actual internal, teacher-intensive pedagogy of the course. It is true, of course, that these other information resources can be used (and often are used) in the same kind of support capacity as textbooks (e.g., audio/visuals which are shown in or out of class but are not particularly important to discussion and do not play a central role in helping students learn key concepts; library assignments which occupy only a peripheral role in a course; drill and practice software which provides for remediation or examination review). As "useful" as some of these applications may be, however, they do not represent true "integration" into the course (any more than most textbooks are integrated), nor do they even begin to realize the potential power of these resources in the teaching/learning process.

Textbooks, then, since they tend to stand outside the classroom, are relatively easily imported into the design of a course, despite the individuality of specific instructors. Likewise, other information technologies, if they are conceived of as occupying the same type of support niche, are also easily adopted (as long as the necessary administrative infrastructure is in place). But to use any of these information resources in a truly active, central, meaningful way, and, especially, to exploit the enormous power of some of the latest information technologies, is not so easily accomplished. The issue is, in Staman's words, not just "curriculum restructuring", but "change in individual courses, one course at a time". Given the enormity of the effort involved in this kind of activity, it is not surprising that IT is not contagious.

CAN IT EVER BE CONTAGIOUS?

We have suggested earlier that to attempt to bring about the integration of IT into teaching and learning without first gaining a proper and complete understanding of why it is not contagious is, at best, foolish, and,

at worst, detrimental to the long-term expansion of its educational use. It is clear, in the light of the discussion above, why such efforts are foolish. If each course, and each instructor, and each teaching/learning environment is unique, and if the best instructional software is that which supports and enhances and extends that uniqueness, then IT developed in one setting simply cannot be easily transferred to another setting. IT is not like the common cold. Successful adoption of preprogrammed software into the internal fabric of an existing course necessarily requires the reworking of that course, or, at least, the reworking of sections of that course. This is not to say that it cannot or should not be done. But it does say that IT will probably never be "contagious" in the sense of enjoying an easy and natural incorporation into existing courses and curricula, and that efforts to encourage its use which do not take into account the specialized nature of teaching and learning probably will not be successful.

This suggests that EDUCOM's Joe Wyatt Challenge, however much good it might have done to publicize the dilemma and to highlight a number of individually successful applications of IT, is not likely to bring about significantly increased use. Joe Wyatt is right. IT is simply not "contagious". Nor are Kozma's "vivid images of change" and "shared success stories", nor even Staman's "critical mass", in themselves, likely to make IT "more" contagious.

But to explain why IT can never be contagious, and why most efforts to encourage its use cannot be successful is not to explain why, in the long run, such efforts may be detrimental to the goal of encouraging higher education to avail itself of the potential value of these technologies. IT in education has come to denote the use of computers in teaching and learning, to the practical, if not theoretical, exclusion of other forms of information technology. This is unfortunate because books and journals and the complex of audio and visual materials, collectively known as educational or instructional media are also part of information technology. And what is impressive about these technologies, in the context of this paper, is that they, like computers, are equally not contagious in the classroom. And in that observation, perhaps, lies a cautionary tale for computerists.

Over fifty years ago, Harvie Branscomb published a book, titled, Teaching With Books (1940). Branscomb's main thesis was that colleges had spent considerable sums in the purchase of books for their libraries, but they were not being used in teaching and learning. What a waste. And now, what could be done to improve this rather dreadful state of affairs? As the history of American higher education shows, nothing much was done to improve that state of affairs. A great hue and cry went up in the 1960's, of course, and much federal money was spent on improving libraries, but not on the demonstrated basis of the need for libraries in teaching and learning. Academic libraries, especially research libraries, have always been an article of faith in the higher educational creed. But the use of libraries has always been marginal and despite recurring cries to integrate them with the curriculum or to make bibliographic instruction a necessary part of a baccalaureate education, little has been done to accomplish either. Why? We think the answer is obvious. Until the use of libraries arises out of pedagogical necessity or more than superficial desirability, they are likely to remain uncontagious. And the financial support and encouragement they received during the sixties will not return, regardless of the pleas of librarians for ever more money to purchase a position for their organizations in the information explosion.

Without belaboring the point, we believe the same case can be made for a/v or educational media, another form of pre-personal computer information technology. Non-computer based educational media was widely touted in the 1960's and 1970's as the answer to pedagogical and curricular dysfunction. Audio and video tapes and sound on slide systems and 16mm films and filmstrips and opaque projectors were all presumed to be crucial to proper teaching and learning and faculty were urged to "get with it" and incorporate multimedia into their courses. But little true "integration" occurred and in many courses a/v simply became a way to decorate a lecture. And money for the a/v based salvation of higher education, once in plentiful supply, became and continues to be harder and harder to obtain in amounts judged sufficient by media specialists. Why? We think the answer is obvious. Until the use of media arises out of pedagogical necessity or, again, out of more than superficial desirability, they will remain uncontagious.

Both libraries and educational media have a presumed importance in teaching. Both have been and continue to be presented as indispensable to learning. Both continue to be urged upon the professoriate as both

solutions to teaching and learning problems and as evidence that a faculty member cares about his students and will offer the variety of learning experiences and opportunities available through the library and media. But, the use of neither is contagious and the use of neither has been shown to be critical or indispensable, generally, for effective teaching and learning. And, we suggest, the continuing and uncritical insistence that computers be used in the classroom risks for computerists the same fate of librarians and media specialists: a marginalization of their special interests and, over time, an increasing skepticism, accompanied by fewer dollars.

WHAT NEEDS TO BE CONTAGIOUS

If information technology in education is not likely to be contagious, if, in fact, it probably cannot be contagious, are we shut up then to plodding along, peddling our computers to a faculty who may like us, but who are not buying much? We do not think so. IT has a large potential, especially IT understood as technology broader than just computers (however central computers may become to our understanding of IT). But we maintain that focusing on making IT (broadly or narrowly conceived) contagious, focusing on simply "increasing its use", is both limiting and pedagogically unsound.

If the heart of an academic institution is the teaching and learning environment, then it is teaching and learning that need to be contagious, and more specifically, the (continuous) reassessment of the acts of teaching and learning in a college or university. Yes, good information technology specialists are calling on their faculty to do this. We have already acknowledged that the most successful efforts to encourage the use of IT are those which involve faculty in extensive examination and redesign of their teaching approaches. But the reasons for doing so tend to be self serving, it seems. The message is, here is IT, it is good and it is important, now, please reassess your curricula, your pedagogy, and your students' learning needs so as to take full advantage of IT. The reason usually given is that these tools may help students learn better, and faster, and more thoroughly, and all of that may be true. But an underlying reason, generally not given, is that, after all, we have spent a lot of money on this and if we cannot justify these purchases on the basis of learning needs we may not get much more. Our language may not be as simplistic, but our skeptical faculties can strip away the IT jargon and get to the heart of our arguments.

But the national calls for higher education to pay attention to teaching and learning have not been prompted by a national desire to use more and better IT. They have been prompted by a growing and substantial dissatisfaction with the products of our schools. And the movement toward active learning or toward mastery rather than mere coverage is independent of the conviction held by information technologists (or librarians or media specialists) that computers in the classrooms will lead to educational reform or that these computers can contribute in some substantial way to educational improvement. What needs to be contagious, what will, in the end, really serve student learning needs, is a full and honest examination of the interactions between learners and information, defined in its broadest sense, in our courses.

We need to assess frankly the manner in which we currently bring about those contacts; we need to ask ourselves what kinds of contacts bring about meaningful and lasting learning; and we need to be careful not to focus on a single information format or technology as the answer to better, or more dynamic, or more relevant contact. The information "within" the student (the web of past experience, perspectives, and understanding the student brings with him) is as vital and as important as the new information a particular educational resource presents. Likewise, the opportunity to examine new ideas collaboratively, both formally and informally, with peers and with teachers, is as important as the opportunity to witness the ideas being discussed by experts from the other side of the globe. And the observation that today's students have difficulty assimilating the critical concepts presented in more traditional information sources (e.g., books and journals) is not sufficient reason to abandon them.

What faculty do need, however, is an environment in which they can reassess their students' interactions with information, an environment in which they are encouraged and supported in their efforts to rethink how their students respond to and understand the information environment in which all learning is embedded. Understood this way, the information environment is much more than the formal information systems on campus.

It includes lecture notes and textbooks, library and audio/visual resources, as well as the knowledge students bring with them to a course or learning experience. And this dynamic reevaluation of a course or courses needs to be conducted in a setting in which there are many information-related opportunities available to the instructor, including a partnership with an information specialist, knowledgable in all information sources, who will thoroughly understand what the faculty member is attempting to accomplish and who will bring his expertise to bear on the problems presented. Leadership, in other words, must come from the faculty; stimulation, new ideas, and technical support must come from the technical experts. In such a way, instructional development can become a full- orbed partnership of pedagogy and technology.

We submit, that faculty and student developed perceptions of the need for better teaching and learning, along with a proper understanding of the relationship of information to this learning, will, with careful administrative support, be contagious. All professionally motivated faculty members want to do as good a job as they possibly can. They want to teach well and they want their students to learn well. Motivated students, likewise, want to learn and prosper academically and intellectually. In such an environment, IT, if it is willing to be a servant and not an arrogant master, will, quite likely, find considerable acceptance. But what is used and how it is used must always be at the discretion of an informed professoriate and not be the result of an effort to find an educational use for yet another piece of computer-driven technology.

BIBLIOGRAPHY

Branscomb, Harvie, Teaching With Books: A Study of College Libraries (Chicago: Association of American Colleges and the American Library Association, 1940).

Staman, E. Michael, "An Action Plan for Infusing Technology into the Teaching/Learning Process", Cause/Effect, 13:2 (Summer 1990), pp. 34-40

"Vanderbilt's Chancellor: a Tireless Advocate for Computer Technology", The Chronicle of Higher Education, XXXVIII:2 (September 4, 1991), p.A28.

Using Hypermedia to Develop a Political Science Simulation

David P. Redlawsk
Department of Political Science
Rutgers University
New Brunswick, NJ 08903
Redlawsk@Zodiac

Abstract:

Political scientists who study voting behavior have typically relied on surveys of the mass public to gain an understanding of how and why people vote. Surveys can provide a great deal of insight into attitudes and behaviors, but they cannot follow the step by step mental processes voters use while trying to make decisions about which candidates to support. An alternative methodology using a computer based election simulation is presented that allows the researcher to track the actual choices made by subjects as they search for information about candidates and make a voting decision. This paper describes the simulation and the hypermedia technology in which the simulation is programmed.

Introduction

For the political scientist who wants to understand voting behavior, the sources of data have traditionally been surveys carried out on the mass public. These surveys, the most famous of which are the National Election Study series carried out every two years by the Survey Research Center at the University of Michigan. These batteries contain hundreds of variables, covering everything from respondents feelings about political figures to their voting preferences to their demographics. The data collected provide a wide range of possibilities for analysis by voting behavior researchers. And yet, because the surveys are carried out at a particular point in time, the data are static, and cannot provide much insight into the ongoing decision processes that voters use in trying to select a preferred candidate out of a field of two or more. Thus, while researchers can readily discern linkages in the data between attitudes and the vote, it is nearly impossible to understand what people think about as they are barraged with information during an election campaign. An alternative means of data collection is needed to get at the ongoing mental processes voters use to make their decisions.

While political scientists have worshipped at the alter of survey research for many years, the questions being asked about the decision making processes of voters go beyond the traditional realm of political science, and into the study of cognitive psychology. Unlike political scientists, psychologists have not embraced surveys to such a degree, preferring instead to use experimental methods to understand questions about individual decision making. If the quest to understand voter decision making is a quest to understand cognitive processes, then it seems appropriate to consider borrowing experimental methods from psychology to apply to political science. The computer based simulation described in this paper does just that. It is an experimental method, presenting to subjects a simulated election campaign, allowing the researcher to record the steps taken by the subject to make decisions between simulated presidential candidates.

This paper does not focus on the justification for such a methodology, since that is to be covered elsewhere.[1] The purpose of this paper is to look at the underlying technology that allows the creation of this simulation - hypermedia - and to discuss how it was used to make this project work efficiently and effectively.

[1] Lau, Richard R., "Information Search During an Election Campaign: Introducing a Process Tracing Methodology for Political Scientists (Working Title)," Paper to be presented at the 1992 Meeting of the American Political Science Association, Chicago, IL.

Hypermedia is the term used here for a wide variety of specific compute products; including Apple Computer's Hypercard, Asymetrix Inc.'s Toolbook, and Borland Inc.'s Objectvision. All of these products, and the others in this area, have in common simple graphical interfaces, script oriented programming, and a metaphor of "cards" (or "pages" in the case of Toolbook) for managing information. Hypermedia also provides simple links between disparate types of data, and gives users the ability to move throughout the hypermedia application with the simple point and click of a mouse.

The Nature of the Project

This project began as a research interest of Richard L. Lau, then at Carnegie-Mellon, now a professor in the Department of Political Science at Rutgers University. Lau's graduate training had been in psychology and included a good dose of experimental methods. His previous work in voting behavior included significant contributions to the use of schema theory[2] in political science, as a way to understand how voters process information. However, the exclusive use of survey data has been limiting. Thus, the idea that these processes could be better understood with an experiment.

Lau began work on this project with a student at Carnegie-Mellon who programmed the initial version of *Ballot Box* using an Apple Macintosh and Hypercard 1.2. This version was used once to run a number of undergraduates through the experiment.[3] The basic requirement was to simulate a presidential election cycle - including primary and general elections. Subjects would see a series of headlines about the various candidates flash across the computer screen. Using a mouse, they would be able to select a headline in order to lean more information about the candidate. A time limit was placed on the campaign in order to place the subjects under some pressure to complete the task. In addition, the amount of information that was to be made available to subjects about each candidate was designed to be overwhelming - too much information for any person to remember and assimilate in the short time allowed. The program was designed to track the choices made by subjects, so that a protocol would be produced at the end showing what information and in what order each subject searched to learn about the candidates.

In order to implement this project, a graphically oriented software program was needed. In addition, the program had to interact with a mouse, since it seemed that subjects would be able to handle a mouse to point and click at parts of the screen more easily than they could any other available input device.[4] In addition, the software had to support jumping around the program - since there was no way to know what particular item a subject would choose, the program had to be able to move through multiple levels of data very quickly, and then to return to the main screen. Further, the program had to be able to record each item selected by the subject and to record the amount of time a subject spent on any particular item of information. Given these requirements, a hypermedia application was clearly called for. So the initial work was done on the Macintosh.

---

[2] Schema theory, as borrowed from psychology, is used by political scientists to help explain how information that is perceived by a voter is processed in memory. Schemas are memory structures which help us to organize our world. These structures consist of nodes representing objects (of varying abstraction) and attributes that are associated with those objects. How we make sense of the world - including the political one - is dependent on the kinds of schemas we have in memory. If, for example, we have a structure for politicians, such that when we think of politicians we associate words like crook, insincere, overpaid, and the like, we are likely to process information about politicians in a manner consistent with these impressions about politicians in general..

[3] Lau, Richard R., "Information Search During an Election Campaign: Introducing a Process Tracing Methodology for Political Scientists (Working Title)," to appear in M. Lodge and K. McGraw, eds., *Political Information Processing*, Ann Arbor: University of Michigan Press, 1993.

[4] A touch screen system might have been better, but was unavailable to the researchers.

The Use of Toolbook 1.5

When Lau moved from Carnegie-Mellon to Rutgers, one of the things left behind was the Macintosh computer. Rutgers political science, to the extent that it used computers at all, was standardized on MS-DOS based machines. Without a Macintosh, the program as written would not run. Further, it had become evident from the first use of the program that some modifications were needed. However, no one was available at Rutgers to do those modifications in Hypercard.

As a graduate student of Lau's, I became involved in the project when I was asked whether I knew any way to convert an application from Hypercard to an MS-DOS format. I investigated the possibilities and found that Toolbook 1.5 from Asymetrix was very similar to Hypercard. Further, a product was available from Heizer Software that would convert about 80-90% of the Hypercard programming into Toolbook scripts. By deciding on Toolbook, we found that we had also added the ability to use color and a great many more pre-defined functions that could be used in our scripts. As it turned out, the use of color became critical to the revision of the system.

In fact, by the time the system had been converted to Toolbook, and a large number of new functions added, little was left of the original design. The basic concepts were similar, but Toolbook allowed us to make major changes in order to better fit our research agenda. And while I was not originally familiar with either Hypercard or Toolbook, the ease of use of Toolbook allowed me to move quickly in the design and re-programming of the system.

Toolbook functions under a hierarchical structure. (Figure 1.) Messages are sent from the level at which they occur on up the structure until a script is found to handle that message. Clicking a button sends a BUTTON UP message to the script for that button. If there is no BUTTON UP handler defined for the button, the PAGE that the button is on will get the message. If it has no BUTTON UP handler, the message will be sent up to the BACKGROUND level, and so on, until a handler is found or an error generated. Every object that is defined on a screen can have a script associated with it. If there is a script, that script will define the events that occur when the user selects that object. If there is no script, the object is generally simply a graphic or a field into which data can be typed. It is the interaction of the scripts, and the messages that get sent through the hierarchy that define how the application operates.

Scripts are programmed in Toolbook using a language called Openscript. Openscript is very English-like in its syntax and use. It may remind programmers of BASIC in its simplicity. However, that simplicity masks a great deal of very powerful function. Openscript has a wide range of pre-defined functions which, when built into a script, allow for the complete control of every object on the screen. As will be seen in the examples below, Openscript is structured, and programming in it is a good test of the ability to master structured programming techniques. This structure is critical to Openscript, as there is no GOTO construct. Everything must be managed through looping structures, calling structures and through the hierarchy of objects.

The Ballot Box Application

The presidential election simulation that we have developed is called *Ballot Box*. It was designed as a research tool to allow us to follow the cognitive processes subjects use in making decisions about supporting presidential candidates. There are two versions of the simulation. The first is the one that descended from the original Carnegie-Mellon design. This is the Scrolling Version. In this version the subject sees headlines which appear at the top of a screen and scroll down it, disappearing from the bottom. The subject selects headlines to see more information about the candidate. Because the headlines scroll off the screen, and because there is a time limit on the campaign itself, the subject is under pressure to choose items of interest. The system tracks all of the headlines that appear (they are randomly generated from a database of over 600 items of information), and records in a log the ones that are selected. This information, combined with the choices made by the subject and with other information gathered from a questionnaire, allow us to analyze the decision making process in

an environment of limited time and limited information. The Scrolling Version is the one that provides the programming examples later in this paper.

The second version of *Ballot Box* is the Information Board Version. This is an attempt to duplicate and extend, using the computer, research done previously by John Herstein in which subjects had as much time as they wished to choose their candidate.[5] The Information Board Version presents an array of boxes on the screen, each labeled with an identifier that indicates the kind of information that can be found by clicking the mouse on that box. The subject has as much time as desired to choose any (or all) of the information available in the system. There is no time pressure and all information is always available on the screen. This version of the system tests the decision making processes of the subject in the best of all worlds - perfect information and unlimited time. We duplicated Herstein's work in order to get a baseline for the Scrolling Version. In addition, by using the computer, we are also able to extend Herstein by adding far more information to the database than he was able to use on a traditional information board. Thus, the Information Board Version of *Ballot Box* is able to contain the same 600+ pieces of data found in the Scrolling Version. And in this version, a subject can take the time, if desired, to look at every one of them!

The use of color became inportant to the experiment when it was decided to use colors to distinquish candidates and information from the two parties. The system is designed to randonly assign either red or blue to each party. Third and fourth colors are used for informaiton that is presented that is either generic in nature - like polls, or which applies to all candidates. Toolbook turned out to handle colors as easily as it handles all other attributes and objects. It was simple to assign colors and to place them appropriately on the screen.[6]

In both versions of the program, subjects first go through a practice session using the 1988 Bush-Dukakis campaign. They spend up to 10 minutes familiarizing themselves with the data presentation format (Scrolling or Information Board) and the use of the mouse to select items. When comfortable with that they move to the actual experiment. The experiment consists of a primary election season, with six candidates - three from each party - and a general election season with two candidates; republican and democrat. All of the candidates are fictitious, but designed to be as realistic as possible in their stands on issues. The goal of the subject is to learn enough about a candidate to vote for the one closest to the subject's own preferences.

The *Ballot Box* simulation is run on an IBM PS/2 Model P70, which contains a 20 Mhz 80386 processor and a large, fast hard disk. An external monitor is attached to this portable machine in order to display colors, since the P70's screen is an orange gas plasma display. After extensive testing it was clear that nothing less than a 20Mhz 80386 processor would be adequate to run the simulation with reasonable speed. The amount of data that Toolbook has to display on the screen, and the speed with which the screen changes must be made ruled against a slower machine.

Examples of the Programming

Two particular screens provide good examples of the power of hypermedia applications. The first (Figure 2) is the headlines screen used in the Scrolling Version of *Ballot Box*. As described above, the six items on the screen are presented in colors, representing the political parties and representing non-specific party information. a New item appears in the top box and all others move down one box approximately every 8-10 seconds. In order to do this, the software must have an endless processing cycle that can be interrupted by the click of the mouse by the subject. In Toolbook 1.5, this is done through the use of an IDLE construct. IDLE

[5] Herstein, J. A., "Keeping the Voters' Limits in Mind: A Cognitive Process Analysis of Decision Making in Voting," *Journal of Personality and Social Psychology*, 40, 1981, 843-861.

[6] Unfortunately, the black and white nature of these proceedings make showing the use of color impossible in this paper. Color turns out, however, to be a very useful tool for our subjects to quickly assimilate party specific information.

operates in the background - any instructions that it holds are executed over and over unless some other activity interrupts it. In this case, the activity that we wanted to always run was the selection of new information to scroll on the screen. The IDLE script (Figure 3) handles this function. A delay counter is built in so that the scrolling function occurs only once every 8-10 seconds. When the counter times out, the IDLE script executes a series of moves that switch each headline on the screen with the one below it, with the bottom one disappearing. The script then calls another script at a higher level (Figure 4) which gets a new headline from the database. This continues until the subject selects one of the headlines using the mouse, or until time runs out for the election.

Each of the boxes on the headlines screen in Figure 2 is programmed as a button, on which the mouse can be clicked. If any one headline is chosen, a BUTTON UP signal is sent to the script that is defined for that button. In the case of the six headlines, each button has the same script, which simply passes to the PAGE level script the number of the
headline. The PAGE level script contains a handler called GETDATA (Figure 5) which takes as its argument the headline number. GETDATA then generates a request to the database, which resides in a different BOOK, to get the appropriate detailed information and to display it on the page SHOWCARD. The subject is then taken to the SHOWCARD page and shown the information. The GETDATA script also records the time and the name of the headline which the subject has selected, using the SEND MARKTIME handler which sends the headline number to a BOOK level handler which records the time in a file.

Once the subject has spent as much time as desired reading the detailed information about the candidate that has been selected, a button labeled DONE can be clicked which will record the amount of time spent on the card and then return the subject to the headlines screen. In this manner the subject can gather information about each of the presidential candidates in the simulation. The subject has a limited amount of time, and time spent reading one piece of information means time spent not reading others. When time runs out, or when the subject feels that enough information has been gathered, it is time to vote for president.

This brings the subject unexpectedly to a screen on which the various candidates are to be rated, prior to actually voting. (Figure 6.) Throughout the simulation, the subject needs only to use the mouse to do everything necessary. This screen had to be designed with that goal in mind, as well. Pre-testing of the original system showed that requiring the subjects to switch from the mouse to the keyboard in order to type in ratings was interfering with the subject's train of thought. So a way had to be found to use the mouse to enter a rating.

Political scientists like to ask subjects to rate people and institutions on a "feeling thermometer" from 0 to 100 degrees. Subjects are expected to use the thermometer to indicate how "warm" or "cold" they feel towards the stimuli. Thus the use of the thermometer graphic on this screen was important to help the subjects understand how to do the rating. Toolbook's scripts provided an easy way to use the thermometer graphic and to allow the subject to simply point and click at a rating. Each rating value (0, 5, 10...) was defined a separate button, with a script which passes to the page level script the button's value. Thus, when the subject clicks on "75", the button passes the value 75 to a higher level script. That script (Figure 7) requests a verification from the subject and then records the rating in the protocol that is being maintained for the subject. Once all candidates are evaluated the subject is taken to the voting screen where one candidate is finally chosen.

Advantages of Using Hypermedia

The most obvious advantage to programming this sort of application in a hypermedia environment is the ease with which it can be done. Because hypermedia is based on objects, each with its own script, it is a simple matter to define basic functions and then to tie them together using the built in hierarchy of objects. In effect, you build a hypermedia application from the most basic unit on up. Scripts for individual buttons can be linked through backgrounds, pages, and books, so that they easily pass parameters amo g themselves, and simplify moving from one screen to another.

Another advantage of hypermedia over traditional programming tools is the use of graphics as an integral part of the application. This allows us to develop applications which are extremely complex from the internal point of view, but which appear extraordinarily easy to use for the user. This was critical to the success of this project, since we intended to have as our subjects people who might have no experience at all with using the computer. A simple practice session to get them used to using the mouse turned out to be all that was necessary for our subjects to have a high degree of confidence in using the system.

It does take some getting used to hypermedia in order to really understand how to use it. For programmers used to the typical program flow of a standard language such as COBOL or FORTRAN, a hypermedia application can be very confusing. There is no one place in which you can see the complete flow of the program, since the program does not have any kind of typical top to bottom flow. Instead, objects are identified and grouped together as needed. Scripts are defined for each object only once, and then re-used at an appropriate place in the application's hierarchy. Consequently, trying to see the big picture of an application can be difficult. Added to this is the fact that both Toolbook and Hypercard do not have any way to print out all of the scripts associated with an application in one pass. Also, the data associated with an application is stored in a special format, within fields that are themselves objects. So, while it is very easy to manipulate the data from within a particular application, creating large amounts of text is better done outside of the application with a word processor. The text can then be imported into the application and placed into appropriate fields.

Perhaps the greatest advantage to using hypermedia to build applications is the ability to tie together not just graphics, but text, pictures, sound, and video in an application. At this point we have done little with this capability, other than to place digitized photographs of the candidates as an additional item of information that can be retrieved by the subject. However, over the long term, the ability to put sound clips, video images, and the like into *Ballot Box* might allow us to improve on our simulation, to make it even more interesting and of more value in our research.

While this project could have been programmed in a more traditional language, like C+ or Pascal, it would have required far greater knowledge on the part of the programmer and a great deal more time than was avaible. Toolbook, using hypermedia concepts, made the initial programming and later modification tasks fairly easy, so that someone moderately skilled in traditional programming could develop, test, and maintain the system within a two months, working only ten to twenty hours a week. Perhaps this is the ultimate advantage of hypermedia products - they allow almost anyone to develop sophisticated systems with a minimum of time and effort.

Figure 1 - The TOOLBOOK Object Hierarchy

At each of these levels there can be scripts that operate as handlers to manage particular functions. Scripts are user written and tell the application what to do when the user has selected the object to which the script is attached. Messages are sent up this hierarchy, from the initiating OBJECT, to the SYSTEM LEVEL, until a script is found to handle the event.

SYSTEM LEVEL	
BOOK	[Hypercard equivalent is STACK]
BACKGROUND	[Unique to Toolbook is the ability to define multiple backgrounds, each of which can contain multiple pages.]
PAGE	[Hypercard equivalent is CARD]
OBJECT (Button, field, etc.)	

Figure 2 - *BALLOT BOX* Headlines Screen

(Note to Carl - figure 2 is a separate page - could not incorporate it in this document.

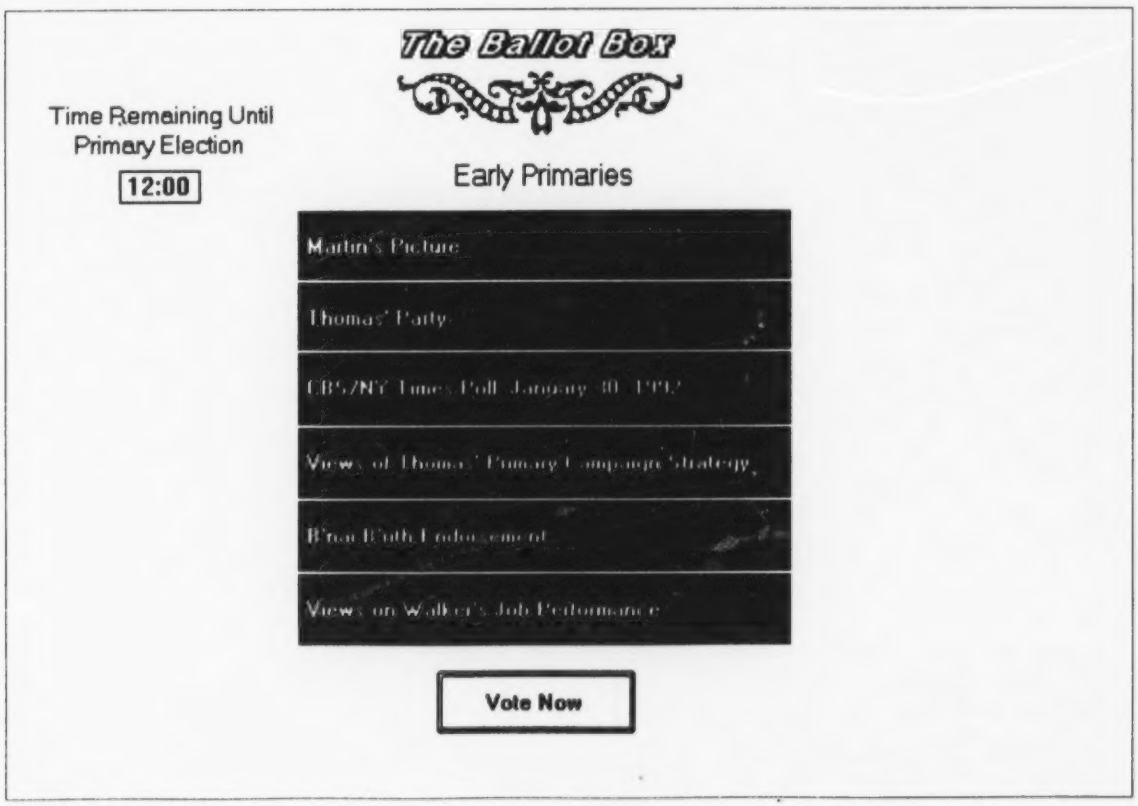


Figure 2

Figure 3 - IDLE Script for Headlines Page

```
to handle idle
if (delayCount < delay) then;                    [This sets the 8-10 sec. delay]
        put delayCount + 1 into delayCount
else
        set syslockscreen to true                [This executes each 8-10 secs.]
        set syscursor to 4
        put 0 into delayCount
        put text of field item1 of page headlines into holdtext1
        put text of field itemnumb1 of page headlines into holdnumb1
        put fillcolor of field item1 of page headlines into holdcolor1

        get newData(it)                          [A seperate script which gets the
                                                  new headline from the database]

        set fillcolor of field item1 of page headlines to word 1 of it
        put word 2 of it into text of field itemnumb1 of page headlines
        put CRLF & " " & words 3 to wordcount(it) of it into text of field item1 of page headlines
        put text of field item2 of page headlines into holdtext2
        put text of field itemnumb2 of page headlines into holdnumb2
        put fillcolor of field item2 of page headlines into holdcolor2
        set fillcolor of field item2 of page headlines to holdcolor1
        put holdtext1 into text of field item2 of page headlines
        put holdnumb1 into text of field itemnumb2 of page headlines
        put text of field item3 of page headlines into holdtext1
        put text of field itemnumb3 of page headlines into holdnumb1
        put fillcolor of field item3 of page headlines into holdcolor1
        set fillcolor of field item3 of page headlines to holdcolor2
        put holdtext2 into text of field item3 of page headlines
        put holdnumb2 into text of field itemnumb3 of page headlines
        put text of field item4 of page headlines into holdtext2
        put text of field itemnumb4 of page headlines into holdnumb2
        put fillcolor of field item4 of page headlines into holdcolor2
        set fillcolor of field item4 of page headlines to holdcolor1
        put holdtext1 into text of field item4 of page headlines
        put holdnumb1 into text of field itemnumb4 of page headlines
        put text of field item5 of page headlines into holdtext1
        put text of field itemnumb5 of page headlines into holdnumb1
        put fillcolor of field item5 of page headlines into holdcolor1
        set fillcolor of field item5 of page headlines to holdcolor2
        put holdtext2 into text of field item5 of page headlines
        put holdnumb2 into text of field itemnumb5 of page headlines
        set fillcolor of field item6 of page headlines to holdcolor1
        put holdtext1 into text of field item6 of page headlines
        put holdnumb1 into text of field itemnumb6 of page headlines

        send changetime                          [A higher level script which changes
                                                  the clock on the screen.]
```

Figure 3 Continued

```
        set syslockscreen to false                [Test to see if time has run out]
        set syscursor to 1
        put 1 into newSeason
        put limit - startTime into duration
        if (sysTime > (duration / 4) + startTime)
                put 2 into newSeason
        end if
        if (sysTime > (duration / 2) + startTime)
                put 3 into newSeason
        end if
        if (sysTime > (3 * duration / 4) + startTime)
                put 4 into newSeason
        end if
        if (sysTime > limit)
                put 5 into newSeason
                set syslockscreen to true
                go to page generalpoll            [If time has run out go to the vote.]
        end if
        if ((newSeason > season))
                put newSeason into season
                if season = 2 then;
                        put "LatePrimaryHeadlines" into whichPage
                        put "Late Primaries" into text of field status of page headlines
                else
                        if season = 3 or (Season = 4 and "Primaries" is in text of field status)
                                set sysLockScreen to true
                                put "Early General Election" into text of field status
                                put "General Election" into text of field election
                                go to page  primariesPoll
                        else
                        if season = 4 then;
                                put "LateGeneralHeadlines" into whichPage
                                put "LateGenCompare" into whichComp
                                put "Late General Election" into text of field status
                        end if
                end if
        end if
end if
end idle
```

Figure 4 - Script to get New Headlines

```
to get newdata str                                  [Gets the new headlines from the
                                                     database using random number generation.]
if season = 1 or season = 2
        put abs (initVal-((random(6))-1)) into getnumb
        if getnumb = 1 or getnumb = 3 or getnumb = 5
                get strokecolor of field "color1" of page headlines
                put "RepCand" into whichcand
        else
                get strokecolor of field "color2" of page headlines
                put "DemCand" into whichcand
        end if
        put random(3) into getnumb
        put " " & textLine (random (textLineCount (text of field (whichcand & getnumb) of page whichPage)))
                of text of field (whichcand & getnumb) of page whichPage after it
        put it into holdhead
        put  "text of recordfield Heading of page" &&  quote & last word of it & quote into stuff
        getremote stuff application toolbook topic databook
        put holdhead & " " before it
end if
if season = 3 or season = 4
        put random (2) into getnumb
        if getnumb = 3
                get  textLine  (random  (textLineCount  (text  of  field  ("Cand"  &  compareVal)  of  page
                        whichComp))) of text of field ("Cand" & compareVal) of page whichComp
        else
                if getnumb = 1
                        get textLine (random (textLineCount (text of field candVal1a of page whichPage))) of
                                text of field candVal1a of page whichPage
                else
                        get textLine (random (textLineCount (text of field candVal2a of page whichPage))) of
                                text of field  candVal2a of page whichPage
                end if
        end if
        put it into holdhead
        put  "text of recordfield Heading of page" &&  quote & last word of it & quote into stuff
        getremote stuff application toolbook topic databook
        put holdhead & " " before it
        if getnumb = 3
                put "300,50,50 " before it
        else
                if getnumb = 1
                        if "RepCand" is in candVal1a
                                put strokecolor of field color1 of page headlines & " " before it
                        else
                                put strokecolor of field color2 of page headlines & " "
                before it
                        end if
                else
```

Figure 4 Continued

```
                        if "RepCand" is in candVal2a
                                put strokecolor of field color1 of page headlines & " " before it
                        else
                                put strokecolor of field color2 of page headlines & " " before it
                        end if
                end if
        end if
end if
writefile ((systime - timemarker) && words 2 to wordcount(it) of it & crlf) to        scrlfile
return it                                                   [The RETURN statement sends the program
                                                             back to the calling script.]
end getdata
```

Figure 5 - Script to get Detailed Candidate Information

```
to handle getData fn
        set syscursor to 4
        set syslockscreen to true
        if "Picture" is in text of field ("item" & fn)
                put last char of text of field ("itemnumb" & fn) into cnum
                go to page ("cand" & cnum & pictures)
        else
                get text of field ("itemnumb" & fn)
                put  "text of recordfield Heading of page" &&  quote & it & quote into stuff
                getremote stuff application toolbook topic databook
                put syserror into errmsg
                put it into text of field heading of page showcard
                get text of field ("itemnumb" & fn)
                put "text of recordfield Information of page" && quote & it & quote into stuff
                getremote stuff application toolbook topic databook
                put it into text of field information of page showcard

                send marktime ("Begin Read Card " & text of field ("itemnumb" & fn) && text of field heading
                        of page showcard)

                go to page showcard
        end if
        set syscursor to 1

end getdata
```

Figure 6 - Feeling Thermometer Evaluation Screen

(Note to Carl - figure 6 is a separate page - could not incorporate it in this document.

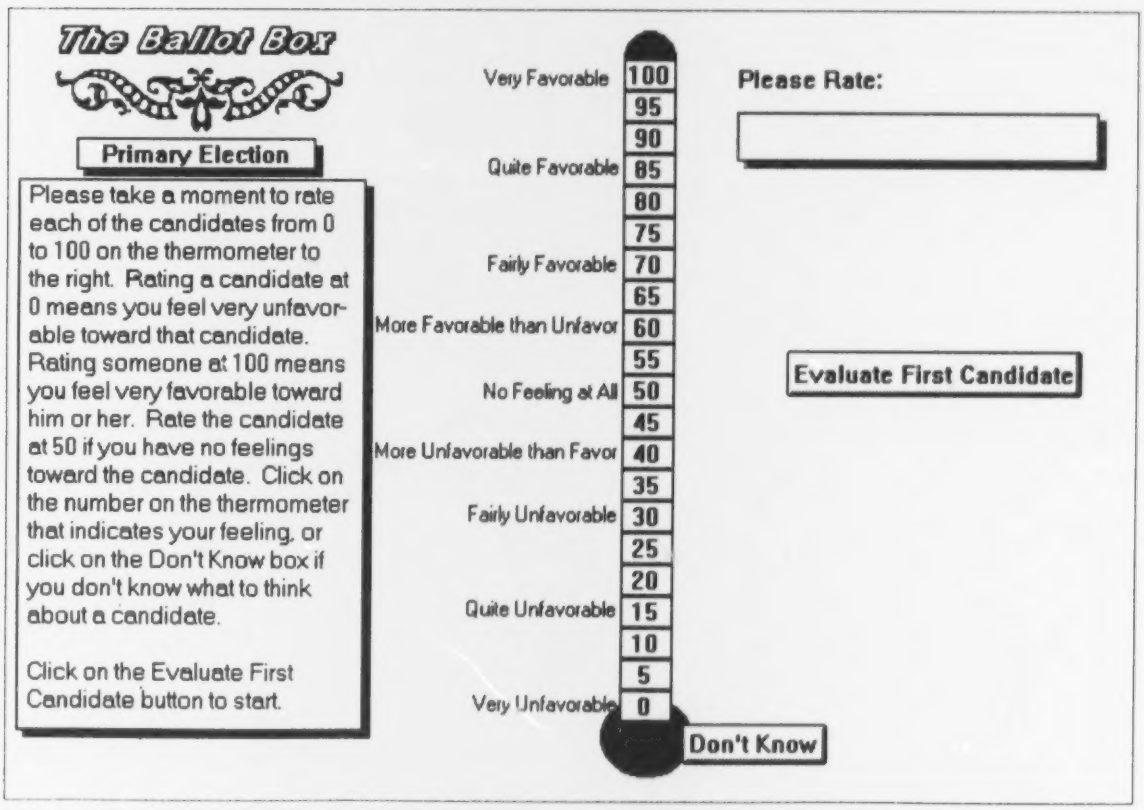

The Ballot Box
Primary Election
Please take a moment to rate each of the candidates from 0 to 100 on the thermometer to the right. Rating a candidate at 0 means you feel very unfavorable toward that candidate. Rating someone at 100 means you feel very favorable toward him or her. Rate the candidate at 50 if you have no feelings toward the candidate. Click on the number on the thermometer that indicates your feeling, or click on the Don't Know box if you don't know what to think about a candidate.
Click on the Evaluate First Candidate button to start.
Very Favorable
Quite Favorable
Fairly Favorable
More Favorable than Unfavor
No Feeling at All
More Unfavorable than Favor
Fairly Unfavorable
Quite Unfavorable
Very Unfavorable
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
Please Rate:
Evaluate First Candidate
Don't Know

Figure 6

Figure 7 - Scripts to Handle Thermometer Ratings

```
to handle thermom bn                          [The themometer's graphic buttons send the
                                               value bn to this script; bn is the rating value.]
system rate
system alreadyrated
if alreadyrated <> "1"
        if text of field cand is not ""
                put "1" into alreadyrated
                put caption of button bn into rate

                send rated (rate)             [This sends the rating to a script which records it in the
                                               subject's protocol.]
        else
                if visible of button firstcand is true
                        request "Click on the button labeled Evaluate First Candidate to begin ratings."
                else
                        request "Click on the button labeled Evaluate Next Candidate to continue."
                end if
        end if
end if
end  thermom


to handle rated rate
system rate
put "You rated " & text of field cand & " as " & rate & ".  Is this correct?" into text of field confirm
show button yes
show button no
send marktime ("Rated " & text of field cand && rate)
end
```

[After this confirmation script runs, the subject either clicks on YES to confirm or NO to try again. Each of those buttons also has an appropriate script which either sets up the next candidate to rate (YES) or provides an opportunity to re-rate the current candidate (NO).]

Using Information Technology for the Remediation of Students in Higher Education

- Integrated Instructional Delivery System -

Man M. Sharma
Clark Atlantic University
Center for Academic Computing
240 James P. Brawley Drive
Atlanta, GA 30314

An Overview

The publication of a report "A Nation at Risk" in April 1983, drove home the point that the educational system of the United States needs major improvements if the nation is to maintain its leadership role in the world, particularly in the technical areas. Unfortunately, the situation has not improved in recent years. Students still enter colleges and universities with less than adequate study habits and less rigorous educational backgrounds, particularly in the fields of sciences and mathematics, than did students of previous generations (McBay, 1990). This was quite succinctly manifested in the latest ETS result bulletin highlighting poor scores in the SAT examination, as compared to the scores of past years.

These problems manifest themselves in poor student performance and high student attrition rates in lower level undergraduate courses, particularly among minority students (NSF, 1989a and b; McBay, 1990). To check-mate this academic performance among students and to reduce their attrition/failure rate, there is a need of taking corrective measures in earnest at the freshman level. Introductory courses need to be seen as being of vital importance. They need to be interesting, comprehensive, and well taught, with an abundance of readily-available help. Unfortunately, there is a general neglect of lower-level undergraduate courses (NSF, 1989a and b). There are three aspects of undergraduate instructional techniques which particularly need improvement: 1. quality of instruction, 2. standardization of course content and evaluation in all sections of multiple section courses, and 3. methods of instruction.

1. QUALITY OF INSTRUCTION: Freshman level courses are often delegated to faculty members who are new, or have relatively little teaching experience (e.g. teaching assistants or sometimes part-time instructors). Consequently, the quality of teaching suffers. To improve the quality of instruction ways need to be devised to involve senior faculty members in designing, developing and implementing effective instructional procedures and materials which can be used by less experienced instructors.

2. STANDARDIZATION OF COURSE CONTENT: Standardization is another problem in introductory courses. In a recent survey of student performance in 3 freshman level courses in mathematics, during a four year period (1986-1989) at one of the HBCU member institution, the percentage of students receiving W, F, I, or D grades in a given course, in a given semester, ranged from 24 to 83 percent, i.e., in one section 24% students received unacceptable (W, F, I, D) grades as against 83% in another section of the same course, and in the same semester. This indicates large variation of student grades, as shown in the following table.

FOUR YEARS DATA FOR MATH 103

	1986	1987	1988	1989
PERCENTAGE-WIFD	55	54	56	55
PERCENTAGE - RANGE	(26-74)	(43-64)	(27-81)	(24-83)
NUMBER OF SECTIONS	5	4	15	11
TOTAL NUMBER OF STUDENTS	142	144	449	593

FOUR YEARS DATA FOR MATH 105

	1986	1987	1988	1987
PERCENTAGE-WIFD	55	71	55	57
PERCENTAGE-RANGE	(32-76)	(55-84)	(19-85)	(22-97)
NUMBER OF SECTIONS	10	9	8	11
TOTAL NUMBER OF STUDENTS	318	262	290	352

FOUR YEARS DATA FOR MATH 106

PERCENTAGE-WIFD	66	94	64	53
PERCENTAGE-RANGE	(44-91)	(86-100)	(60-100)	(40-60)
NUMBER OF SECTIONS	5	3	3	3
TOTAL NUMBER OF STUDENTS	161	80	102	122

It is hard to say which factors contribute to this variation. It may be that some teachers are too demanding, or are absolutely unsympathetic to students. On the other hand, it may be that some teachers are too lenient in grading or are over-sympathetic and less demanding. Certainly, some variation is due to random assignment of students to sections, but much of the variation seems to be due to differences in the quality of instruction. Inexperienced instructors tend to emphasize areas in which they feel comfortable while skipping or giving only cursory attention to other areas of a course and then cover up the criticism of students by inflating grades. Consequently, course content often varies significantly, among different sections of the same course. Furthermore, this uneven classroom coverage is often reflected in an uneven test-coverage, resulting in misleading student grades, large attrition, and hidden-gaps in student preparation, which in most cases do not become evident prior to enrollment in more advanced courses. There is a need to standardize course content and evaluation procedures in order to alleviate these problems, namely, large variation of grades, and heterogenous teaching practices.

3. METHOD OF INSTRUCTION: Much of the content of the freshman level courses is not new to most of the students taking these courses. Similar courses are often taught at the high-school level or even earlier. The major cause of failure is the way in which the matter is presented. The standard course plans with infrequent tests and liberal attendance policies do little to help students in developing habits which involve regular study and class attendance. Little purpose will be served by presenting the same content, once more in a similar fashion and that too at an accelerated pace --- i.e. a course covered at school in one year is covered within a quarter or a semester in college. The method of instruction, therefore, must be interesting and challenging in order to improve student motivation, attention span, and study habits.

The three identified problem areas will be addressed directly in the proposed system. First, faculty members involved in developing the system are experienced faculty members who have demonstrated adequate commitment to improve the quality of instruction. These faculty members have developed materials and procedures which can be used by less-experienced instructors. Variation in the standard of teaching in different sections of the same course can be controlled by effectively coordinating teaching by increasing the frequency of common exams, and also by providing orientation to instructors for using the methods and materials designed by experienced teachers. The method of instruction will be designed in a way which will promote regular class-attendance, and increase student attention span by making the lectures dynamic and interactive. It also seeks to involve the students in several instructional activities by providing grade incentives.

The three identified problem areas will be addressed directly in this paper. First quality and methods of instruction can be improved through Electronic Lecture Notes and Classroom Presentation Options produced by using object-oriented hypertext environment provided by the "ToolBook" as discussed below. Secondly, in order to insure standardization, and comprehensiveness of the course content, a system called Integrated Comprehensive Instructional Support System (ICISS), described below, can be integrated in the instructional

process. Combination of Electronic Lecture Notes, CPO of IBM, and the ICISS is termed as Integrated Instructional Delivery System (IIDS). The IIDS can help alleviate the problem discussed above as indicated by several pilot studies.

Re-designing the existing courses and to make extensive use of technology for alleviating the problems stated above is the primary objective of the proposed system. There is a need to provide structured/regulated mode of instruction. The IIDS includes: frequent testing, electronic lecture notes, electronic grading and record-keeping, extensive reviews and practice testing, self-diagnostics, electronic tutorials based on diagnostics and periodic progress reports. The IIDS (described in the next section) will also make classroom sessions more interactive by incorporating electronically projected questions in which the entire class responds via keypads.

There are two major components of the IIDS: the first component seeks to enhance classroom instruction by using a daily quiz format as well as electronic lecture notes, produced by using latest object-oriented hypertext technologies and is therefore available only for a few courses.The second component aims to provide effective support to classroom instructions in the form of extensive practice testing, lesson reviews, make-up exams, and periodic students' progress reports. The second component is already being used in more than 60 institutions of the MSIP/CAU consortium for several courses in social and physical sciences with significant positive outcomes, as discerned in some of the studies described elsewhere in this paper.

Goals Underlying the IIDS

The primary goal of the IIDS is to reduce the attrition rates and increase the comprehension level of students enrolled in the beginning level undergraduate courses. The strategies we propose to use are to involve students in a variety of instructional activities by linking them to the assignment of course grades, since course grades are what students care most about. There is no substitute for hard work for students to succeed or excel in mathematics. Therefore, the whole instructional system must be such that: 1) the course plan includes a variety of activities that are directly linked to assigning course grades; 2) extensive daily assignments are given that are tied to course grades, and 3) classroom teaching is made more lively, dynamic and interesting.

Description of the IIDS

The IIDS involves seven major differences from the traditional instructional methods followed in any undergraduate course. These differences are made possible by the use of the Integrated Comprehensive Instructional Support System (ICISS), a software package described in the ICISS section of this paper, IBM's Advanced Academic System and Classroom Presentation Option, and the availability of the Electronic Classroom (EC). Under the proposed model, a highly structured instructional format will be followed. The major differences from the traditional instructional format are described below:

1. **Daily Quiz.** The daily quiz system promotes study habits and regulates class attendance. This component of the model has several benefits for students. First, they receive the benefit of immediate feed-back. Secondly, the results of these tests which are stored on the network are immediately available to the instructor. A network security system will assure the confidentiality of the results that may be directly transferred from the result file into an electronic gradebook. Under this model, punctuality and attendance in every class session becomes mandatory because the first 5 to 7 minutes of every class session consists of a quiz, the results of which are included in the overall class grade. The ICISS (see pages 69 and 70) and a large question bank with hints and solutions will be the main instruments to be used for implementing this component of the IIDS.

2. **Mid-Lecture Discussion using the AAcS.** Several studies have shown that even some of the best teachers cannot hold student attention for more than 10 minutes in a continuous lecture. Increasing attention span is crucial for understanding the concepts presented by the teacher. Using the AAcS, lectures will be interspersed with discussion questions to be responded by students via response pads.

After every 7 to 10 minutes during the lecture session, questions will be flashed out for discussion. Electronic Lecture Notes containing a comprehensive discussion of topics incorporating graphics and animation will be usedwherever appropriate in order to enhance the quality of class discussions during each of the 7 to 10 minutes segments of a lecture, followed by electronically projected questions. The AAcS question banks, and Electronic Lecture Notes (ELNs) will be the main instru ments to implement this component. Electronic Lecture Notes (ELNs) will be prepared by using object-oriented hyper-text authoring system (ToolBook), and the Classroom Presentation Option (CPO) of IBM's Advanced Academic System (IBM-AAcS).

3. **End of Lecture Review.** Recapitulation of the entire lesson by answering a few questions is a means of pulling threads together and for providing an impetus to the forthcoming lectures. It is in this phase of recapitulation provided by the lecture quiz on the terminal, that students will be required to answer questions at the end of the lecture hour. This method will help in holding the attention of the class until the very end.

4. **Liberal Computerized Practice and Review.** The Electronic Classroom will function as a computer laboratory for students willing to take computerized practice quiz or tutorials, during the time it is not scheduled for class. Students will learn about their deficiencies from the diagnostics provided at the end of practice tests and will be provided the necessary prescriptions.

5. **Incentives for Computer Laboratory Use.** The course plan will include grade incentives for increased computer laboratory use. The Record Management System (RMS) program of the ICISS provides reports on individual student performance, and time spent in the laboratory.

6. **Use of Electronic Classroom.** This will modernize classroom presentations by administering a quiz every day, instant grading, instant individualized error analysis, automatizing record-keeping, response to discussion questions from every student, response to the end-of-lecture review from every student, and individualized up-to-date student progress reports.

7. **Computerized Record Keeping and Progress Reports.** At present, many instructors use spreadsheets or computerized gradebooks to facilitate record-keeping. Two major features of the Course Management System (CMS) program of the ICISS are not commonly found in existing systems. First, the CMS has the ability to directly import scores into the gradebook from an ICISS result file (files that store results of student scores for tests taken on the terminal) and/or machine-scored answer sheets. Electronic transfer facility saves time for the instructors, and also reduces the time required between a test and the distribution of up-dated feedback.

We believe, that the proposed technology-based model provides a classroom where there is a high probability that students will be reinforced for emitting learning behavior. The foremost assumptions in the proposed approaches are similar to what underlies B. F. Skinner's analysis that behavior is controlled by environmental contingencies of reinforcement. The point is that the teacher, who is an important part of the environment is in a position to control student behavior by manipulating contingencies of reinforcement, so that randomness in learning that typifies many classrooms is eliminated.

Software Support for the IIDS

There are three major systems that will be needed for the IIDS; 1) the Advanced Academic System and CPO of IBM; 2) the Electronic Lecture Notes available from CAU, Institute for Academic Technology and University of Maryland at College Park; 3) the Integrated Comprehensive Instructional Support System (ICISS), already available and in use at CAU and the member institutions of the MSIP/CAU consortium. These systems are described on the following page.

ICISS

The Integrated Comprehensive Instructional Support System (ICISS) was developed jointly by Dr. John Gillam (Ohio University), Dr. Donald Norris (Ohio University) and Dr. Man M. Sharma (Clark Atlanta University). Originally developed as Math Editor (MEDIT), a scientific editor allowing the production of material involving scientific expressions and symbols, the ICISS has evolved into several integrated programs. These programs and their functions are listed below:

1. **MEDIT:** It is a scientific editor used for producing documents, question banks, and assignment sheets that may involve the use of scientific expressions.

2. **TESTGEN:** It is used for producing multiple versions of class tests with the same degree of difficulty, or practice, and review tests to be taken on computer terminals.

3. **TERMTEST:** It is used by students to take regular and practice tests on the computer terminal. The program offers two choices to students - to either review the test-related material with immediate assistance in the form of hints and solutions, or to take a Simulated Timed Terminal Test (STTT). After completing an STTT, the program provides several other options, including reviewing the solutions of items missed on the test, and/or viewing diagnostics of test performance and the instructor's advice for improving areas of deficiency. All information about the testing session of a student is saved in different files, including a result (RES) file containing detailed information on the student's performance, and a statistics (STT) file containing information needed for item analysis.

4. **RECORD MANAGEMENT SYSTEM (RMS):** The RMS performs three functions: 1) cleans the result file of extraneous records, 2) produces raw reports of student activities in a user-defined format, and 3) provides summary reports combining multiple records of a student in a single record.

5. **ITEM ANALYSIS (IAN):** The IAN provides information about the items used in Simulated Terminal Test, descriptions of those items, the number of times each item was attempted, and the number of times each item was responded correctly. This program also provides an analysis about the academic deficiencies of the whole class as well as drawbacks of individual students.

6. **COURSE MANAGEMENT SYSTEM (CMS):** This system provides a computerized grade book which includes the ability to import scores directly from the result file, or from Scan-Score answer sheets. In addition, it provides individualized progress reports for students.

APPLICATIONS

The ICISS was originally designed in 1980 to alleviate the problem of remediation in mathematics at Ohio University. Since then, this system has gone through several modifications and enhancements, has proved to be very effective in reducing attrition, and improving the achievement levels of students at several institutions in a variety of freshman-level courses. (see next section on some case studies.)

The system is like a "shell," in the sense that it can be used for any subject area by just appending the appropriate question bank. Because of the generic nature and flexibility of the ICISS, it is being used at present for 24 different courses as listed below, and several other courses are at the beta testing stage.

Basic Math	Micro Economics
Introductory Algebra	Macro Economics
College Algebra 1	English Composition
College Algebra 2	Scientific Report Writing
Calculus 1	Physical Science
Calculus 2	General Physics 1

Introductory Statistics
Business Mathematics
General Biology
Introduction to Botany
Micro-Biology
Earth Science
General Chemistry
Intro. to Computer Application
Fortran Language
General Physics 2
General Psychology
Introductory Sociology
GRE - Verbal
GRE - Quantitative
SAT - Quantitative

The ICISS provides effective supplementary support to classroom instructions. Before each class test, instructors can design reviews and practice testing sessions for students, using already available large question banks with hints and solutions/explanations. Several documented studies have established that this support helps students to:

* regulate study habits,
* minimize test anxiety,
* be informed of their level of preparation for scheduled tests, and
* view instructor's advice for problem areas.

This system facilitates instructors' efforts to institute liberal make-up policies for students with genuine excuses, and provides periodic individual progress reports.

There are several ways ICISS can be used and is being used effectively to support classroom instructions. The key factor in implementing the ICISS is the availability of a large question bank with hints and solutions. Fortunately, we have these question banks for over 24 courses, at freshman level, listed above. These question banks were produced and beta tested by over 100 faculty members from more than 60 minority institutions under projects supported by the MSIP, U.S. Department of Education, and IBM Corporation. Some of the commonly used practices for the ICISS are incorporated in the design of IIDS as discussed earlier.

Some Studies in the use of the ICISS

There are more than 60 institutions using the ICISS for different subject areas. Several of them have conducted studies to investigate the effect of using the ICISS adopting different instructional formats. Some of these studies are mentioned below.

1. Donald Norris, Ohio University
   (White American Students, 1981)

   In this study twenty-four sections (715 students) of an Introductory Algebra course were exposed to a regimented format of instruction involving daily quizzes, frequent testing, and weekly progress reports using the ICISS. Students averaged one letter grade higher than the earlier performance of students using traditional approaches. In addition, the attrition in the next-higher-level mathematics course dropped by 17 percent. Ohio University is still using the same approaches in lower level mathematics courses.

2. Eugene Schuster, University of Texas at El Paso
   (Mexican American Students, 1987 - 89)

   A similar format to that of Ohio University, without daily quizzes was used in the Pre-calculus (370 students) course. A 13 percent reduction in the attrition rate was observed during the Spring 1988 semester as compared to the attrition in the same course during the spring semester of 1987.

3. Joseph Coble, Clark Atlanta University
(Black American Students, 1987 - 90)

Dr. Coble of the Department of Psychology has been using the ICISS system, since the fall of 1987. He has used the system for both classroom tests and computer-terminal practice tests, comparing the classroom-test performance of High Lab Use (the top 1/4 of the students in terms of lab use) and Low Lab Use (bottom 1/4) students. The High Lab Use students performed significantly better (more than one letter grade higher) than Low Lab Use students. Furthermore, Dr. Coble concluded from his data that most of the benefits from use of the system are due to learning of course information rather than memorizing specific items.

4. Delores Reed-Sanders, Pan American University
(Mexican American Students)

This study illustrates an attempt to integrate microcomputers, using the ICISS in an Introductory Sociology course. Computer integration was organized around general learning objectives based on substantive areas augmented with tutorials of test items with hints and solutions, and practice tests. The computer integration was evaluated with an experimental design which compared traditional versus computerized introductory Sociology courses. Findings showed the computerized group excelled beyond the traditional group in terms of performance and attitudes. Furthermore, this study suggests that students of all ability levels benefit from computer integration and utilization. (Abstract of paper presented in the MSIP Conference, Nov 2-4, 1988).

In summary, the development of ICISS has most definitely been user-driven. The original program was developed to meet a specific problem in mathematics and science courses, and subsequent programs have all been aimed at solving specific problems of students and instructors. The ICISS software package is certainly not finished--it will likely never be finished--as is the case with other widely used programs. New programs are currently under development to meet some of these problems, and the future will certainly see more new programs. The ICISS has proven to be an extremely useful system, in terms of both user (students and instructors) satisfaction and has also demonstrated improvements in students' academic performance.

The Advanced Academic System (AAcS)

Based on the power of the IBM PS/2, the AAcS is designed to make lectures, papers, and course materials easier and more effective. Now a varied assortment of multimedia capabilities and functions have been added to further enhance these processes.

The AAcS consists of four Personal System/2 configurations, pre-loaded and packaged with "best-of-breed" tools and academic sample applications. An optional feature is the new, improved version of the Classroom Presentation Option, a kit which includes sample software and 24 student response keypads. The Classroom Presentation Option creates an interactive classroom environment to enhance the learning and lecture processes. The Advanced Academic System (AAcS) of IBM includes object-oriented hyper-text authoring system - ToolBook. The system can also be interfaced to the Classroom Presentation Options (CPOs) which includes a set of student response keypads.

Most importantly, the AAcS delivers multimedia capabilities over a wide range of hardware platforms starting with the powerful Model 80 which comes preloaded with multimedia features, and extending to the sophisticated Model 70 and Models 57 and 55SX. For added flexibility, this full-featured multimedia solution is offered in a tower version (Model 80) and a DESKTOP version (Model 57). All systems are multimedia enabled. Thus, a user can start with a full function developmental system (Model 80) or begin with the Model 70 or 55 SX and add optional features and devices expanding multimedia usage and sophistication. The forthcoming updated version of IBM's AAcS with Digital Video Interactive (DVI) features will open enormous

opportunities of storing motion video scripts on the hard disks, enabling the use of video frames in the course-work.

Electronic Lecture Notes

The contents of most lower level courses in science and mathematics have a standard sequence of topics as given in textbooks. This standard sequence is used to prepare electronic books made much more livlier and dynamic than the printed materials by integrating the use of graphics and animation. Intent of the Electronic Lecture Notes is to capture the imagination of master teachers and incorporate their style and pedagogy using graphics and animation, wherever necessary, for the benefit of less experienced instructors. The use can be made either for the entire lecture or for only those topics where it is difficult to teach/learn by traditional "chalk and talk" method. Moreover, these notes can be loaded in the computer network for the use of students.

Another advantage of these Electronic Lecture Notes is to make classroom teaching more interactive. We all know, that a good lecture must always involve students to respond to questions related to the topic of discussion. But traditionally the response generally is only from a few "more confident" students. In Electronic Lecture Notes used with Classroom Presentation Option (CPO) of the IBM's Advanced Academic System (AAcS) the entire class can be involved to respond to a question. The question is electronically projected on the screen, the students respond to the question using response pads (comes packaged with CPO of AAcS), the response analysis appears on one of the windows of the question screen. This response analysis provides continuous assessment of classroom teaching. Additionally, individual student responses are saved in a separate file, that provide basis for the teacher to identify students who need counseling/advising.

Experienced master teachers, with assistance from educational technologist, design and prepare scripts with suggestions for graphics and animation wherever appropriate. These scripts are then programmed by experienced programmers. Using programming with TOOLBOOK is relatively much easier especially if the scripts involve graphics and animation. Moreover, once the product is complete, it is as easy for the instructor to make modifications as pressing a function key and typing over the changes.

The IIDS Overview

Target Population: College freshman and high schools

Target Courses: Basic Skills and beginning level undergraduate courses.

Focus: Improve classroom teaching; provide infrastructure; reduce attrition and improve performance.

Methodology: Conduct classes in electronic classrooms equipped with projection system and the AAcS of IBM.

- Follow structured lesson plan
- Use non-linear hypertext environment to make lectures dynamic
- Use Classroom Presentation Option to make lecture interactive
- Make maximum use of classroom time by projecting and discussing instead of writing and drawing
- Provide extensive out of class support for lecture reviews
- Provide periodic progress report to students

Infrastructure:

- Networked laboratory equipped with IBM's LANKIT or DESKLAB
- AAcS and Classroom Presentation Option with keypads
- ICISS and question banks
- Electronic Lecture Notes and CPO question banks.

Implementation Recommendation

The three most important parameters of the IIDS, or for that matter any curriculum reform, are: 1) strong administrative commitment and support; 2) building consensus among faculty and providing ; and 3) developing local model on a smaller scale upon which to build the curriculum reform on a wide scale.

1. The administration must appreciate the importance of the reform and allocate necessary resources to implement the system. Key person(s) must be identified to lead the effort. These persons must be knowledgeable, experienced and dedicated teachers, and well respected among their colleagues.

2. The system should be thoroughly discussed with faculty through seminars, and persons already using IIDS should be invited to demonstrate the system to discuss its rationale. After building local consensus, basic infrastructure should be built in the form of a pilot electronic classroom, and faculty should be trained in the use of the IIDS, and to serve as course coordinators. It is important to involve at least one senior experienced faculty for each discipline area.

In order to implement the IIDS with all its components we equire an electronic classroom with networked computers, and an electronic projection system, in addition to the needed material and software.

One may argue, that since most institutions have a problem in providing adequate space even for regular classes, where could they find additional space for electronic classrooms? The concern can be redressed, as at Clark Atlanta University, by converting regular classrooms into electronic classrooms, using specially designed NS/2 workstations. Classrooms can be used as regular classrooms or electronic classrooms depending upon the instructional design.

3. Already available materials in the form of question banks and Electronic Lecture Notes should be modified to accomodate local needs. Pilot studies should be conducted to streamline the procedures and to establish local models for one or two courses. Seminars should be conducted to discuss the results. These experiences should provide the basis for implementing the IIDS at all the lower level courses.

It may be argued, how can teachers take upon themselves additional load of learning new technologies and changing the conventional instructional procedures, specially when they are already saturated with the work? The design of IIDS replaces the traditional routine type of instructional tasks like grading, record keeping, attendance, advisement with automated system and it is expected, that the time thus saved can be used for learning new technologies and its use to enhance classroom instructions.

Conclusion

The extensive use of technology in the IIDS should not only prove beneficial to students, but also allow more participation of senior instructors, and provide effective tools to less experienced teachers for improving the quality of teaching in introductory courses. Teaching lower level courses, in the traditional way using the "Chalk and talk method," demands more time and energy from the teacher as it involves providing special attention to each and every student. Fortunately, the use of technology involved in the IIDS will allow more involvement of senior instructors without placing extreme demands on their time. Moreover, once procedures and materials, are developed, beta-tested, and implemented by senior instructors, they can be used by other instructors after proper orientation. If standard course material and plans are available, and if instructors have access to and training in the use of appropriate technology, then materials can be readily obtained and procedures can be implemented, without placing undue demands on the time of instructors.

Computer-mediated communication in two curricula: Exploring a telecommunications literacy

Matthew Stuve
Joan Rose
Donna Satterfield
Franklin College
Franklin, IN 47907

ABSTRACT

A new goal for academic computing is to teach students to be participants in the Information Age, not simply passive users. To this end, Franklin College has adapted two courses from two distinct curricula to include networking and telecommunications activities. The first course, a computer literacy laboratory, utilizes the concept of functional learning to teach students about working in a networked environment. The other course, a seminar for seniors in education, employs a discovery learning approach to inform students on the application of telecommunications to their teaching careers. Students in both courses gain practice with computer-mediated communication (CMC). This is an informal study that evaluates the success of applying networking and telecommunications in low-cost instructional settings. The observations from both classes will help faculty prepare future instructional applications that go beyond the literacy paradigm, which emphasizes computer tools, to explore the social and collaborative aspects of computing. From these experiences and other trends in computing technology, components of a telecommunications literacy are suggested.

THEORETICAL OVERVIEW

Telecommunications networks have been shown to foster new opportunities for exploring many instructional domains. During the 1980s, the advent of the personal computer made possible affordable, localized access to the large mainframe computers of various research institutions and state governments. As a result, with only a computer and a modem, any classroom can benefit significantly from the technology of telecommunications by dialing up to these remote computers. Telecommunications empowers students to explore external sources of current information and discover for themselves the value of the technology to their careers.

Similarly, by using electronic mail in a networked laboratory, students can continue class activities and instructor correspondence well beyond the scheduled class hours. What is intriguing is that most of the learning that occurs in these situations is student initiated, internally motivated, and usually incidental to the actual assignment. Applying telecommunications in this manner can be thought of as functional learning (Cohen & Miyake, 1986; Mehan, 1989). Learning is functional when it has purposes separate from the contrived instructional activity. According to Cohen and Miyake, "the functional learning environment is based on the assumption that the most powerful learning occurs in a situation where the learners are aware of what should be learned, how, and for what purpose." (1986, p. 258,).

Both applications, remote access and localized networks, foster computer-mediated communication (CMC) (Bannon, 1986). CMC embodies various technological solutions to the problems of augmenting student-instructor and student-student communication. CMC can be implemented with modem-based, terminal emulation technology, i.e. dial-up services like *Prodigy* and *CompuServe*, or it can use high-speed computer networks based on mainframe computers or simple file servers. Examples of the latter can range from local area networks (LANs) to the Internet itself. Both methods can utilize electronic mail.

The mode in which students encounter CMC can be of two types: asynchronous and real-time. Asynchronous communication simply means that the correspondents send concise messages alternately. Most e-mail services have disk space allotted for such messages to reside until the receiver is ready to read them. Real-time CMC,

refered to as "chatting", is the ability for both correspondents to conduct a written dialogue on the computer screen simultaneously. In this project, we will make use both methods of networking and both modes of CMC.

PROJECT GOALS

In Blaschke (1988), Margaret Riel defines four essentials for successful networking: 1) there must be a task to accomplish; 2) the network must be easier and more effective than other means available; 3) participants must know each other; and 4) some degree of accountability must exist. Activities that use these guidelines will stand a greater chance at maintaining the interests of the students over a longer period of time. With this in mind, the following questions guided our implementation of CMC:

1. How can we teach students to participate in on-line discussions and gain appreciation for computer-mediated communication?

2. What skills and attitudes are essential for students to pursue telecommunications on their own?

3. What instructional and interpersonal changes occur in a learning environment in which the instructor and students are empowered with network tools such as electronic mail and telecommunications?

4. What telecommunications and networking experiences are most meaningful to students and how can we use these prepare students for their careers?

We pursued CMC in two courses. One is an entry-level computer literacy course, CMP 130, which introduces freshmen and some upper-level students to the Franklin College Computing Network and the essential tools of computing. The other, EDE 455, is a senior-level course for education majors that addresses issues and problems in education. In both courses, the instructional goals were achieved without incurring additional hardware expenses. Faculty and computing center staff worked together to facilitate low-cost solutions that met the instructional goals and permitted monitoring of the students' performance. CMP 130 used an existing laboratory with a VAX-based network environment. EDE 455 required reconfiguring existing hardware and software in a library resource center.

Evaluating these two courses relied upon theories and methods described by Cohen and Miyake (1986), Riel and Levin (1990), Levin, Kim, and Riel (1990), and Stuve (1991). Our evaluation focused on the role of the technology, the effects on student-teacher communication, and the effectiveness for instructional applications.

Functional Learning in a Networked Environment: CMP 130

CMP 130, Computer Literacy, is designed as an introduction to computing and explores the impact computers have on society. Students practice fundamental programming concepts and gain hands-on experience in using application software. The course places emphasis on the role of computers in business and is, consequently, a required course for those majors. This course met in the lab three times a week for 50 minutes. There were 12 women and 11 men enrolled in the course.

CMC was used in this course not only for demonstrating networks, but also for integrating all of the course computing activities into a common, non-restricted, environment. That is, from any network station, students could check their mail, carry out their assignments and submit them to the instructor.

Learning Activities

In addition to individual word processing, spreadsheet, and database assignments, students were organized into groups during their major class project. These projects were designed to require extensive collaboration between the group members so as to necessitate asynchronous communication and common file access. This approach demonstrated a functional learning model of instruction because the students learned about working in a computing network not only from instruction provided in lectures, but by participating in a representative environment for the duration of the course. The functional purpose, in the case of CMP 130, is to promote open communication between the instructor and the students and among the students themselves. In other words, students were provided with a learning environment in which they had to build communication links for their own purposes. In the context of their major project, students used CMC to coordinate the activities of the group.

Since all activities were based on the VAX network, CMP 130 became, for the most part, a "paperless" course. Working in cooperation with the network systems manager and the staff of the computing center, the instructor was able to adapt her course activities to the capabilities of the VAX network. One member of the computing center staff was available on-line to answer technical questions from the students and the instructor.

Previous to this semester, quizzes were administered during the instructor's office hours. With this network implementation, quizzes were administered so that the students could complete them at any time during the semester. Once ready for a quiz, the student would send a message to the instructor to receive the quiz as a binary attachment to an electronic mail message. Once completed, the student would then send it back using the same procedure. This method required multiple versions of the same quiz so that students who took the quiz earlier did not share information regarding quiz content. Six versions for each of eight quizzes were produced during the semester. The benefit of this procedure was that it freed up more office hours for general consulting.

Some of the e-mail learning objectives included the following tasks: sending and replying to simple messages; text transfer; binary transfer; message forwarding; annotation and relaying messages; archiving messages; creating remote folders; purging remote folders; using distribution lists; and carbon copying.

Observations

Overall, the effect of using CMC significantly increased the opportunities for communication between the instructor and students. Out of 23 students, 20 became reasonably proficient with the electronic mail system. Ten of those became prolific communicators with the instructor (43%). The remainder of the active group used e-mail primarily for submission of assignments. Discussions between student and instructor were primarily topical responses to issues brought up in lecture or questions about assignments.

These results compare quite favorable to the number of active communicators in last semester's non-networked version of the same course. In that course, only four or five students from a class size of 32 students (approximately 15%) communicated regularly with the instructor during non-class hours. The CMC implementation shows almost a 300% improvement in number of students involved in regular, non-class communication with the instructor. This is the most important and welcome effect of using a computer network to facilitate communications between the instructor and the students.

The number of transmissions received by the instructor ranged between 30 to 60 messages a week. The number of daily messages increased to an average of 20 messages on days preceding due dates of assignments. Most messages were responded to within 24 hours.

The purpose of the instructor's messages was three-fold: 1) Provide encouragement and personal consultation to the students, either unsolicited or as a result of a student-initiated dialogue; 2) Provide relevance to classroom discussions or clarify complex concepts; and 3) Provide explicit instructions to tasks encountered during non-class activities.

Since the purpose of using CMC was to augment--not replace--interpersonal communication, it was ironic that the five most prolific e-mail users very rarely visited the instructor in her office. Those that did visit in person were able to go over problems on the instructor's computer. E-mail assistance to a simple task such as printing a spreadsheet on a network usually consisted of a list of steps or "recipe". Expressing solutions to student questions in the form of a list required more time on the part of the instructor, but for many problems, this was the preferred method of assistance, especially by the prolific group. However, higher order intellectual tasks, such as analyzing spreadsheet data in preparation for a graph was somewhat more difficult to express in an e-mail message.

Conversely, the students who were infrequent users of e-mail made use of office hours. One explanation may be that those students who used e-mail extensively had previous computer experience which made them more confident with the technology. This was not the case because from initial class surveys all of the students identified themselves as computer "illiterate". The experiential, academic, and gender attributes were evenly distributed among both groups. Another explanation may be that the students in the prolific group had communication skills more suitable for writing. This study did not analyze this hypothesis. It is unclear why some students chose to make more extensive use of e-mail in general. The instructor attributes this distinction to student personality and his or her compatibility with other students and/or the instructor herself. Overall, however, there was no difference in student performance across all instructional objectives between the prolific group and the occasional users of e-mail.

The overall impression from the instructor was that the network provided a non-threatening means of communication. That is, students who may have felt intimidated by face-to-face communication with the instructor, or embarrassed to address the entire class could use e-mail to facilitate some degree of communication. This does not suggest that e-mail relieves these communication anxieties but rather it initiates a limited dialogue that may not have occurred at all in a traditional classroom. By providing support via e-mail, it was hoped, the instructor was able to encourage the student to become more active in classroom dialogue. Furthermore, incidental communication activities evolved which improved learner confidence with the environment. Because of the network environment, the students learned the how to participate in the culture of computing, where all tools and, more importantly, all people were readily available by means of CMC.

Instructor Workload

The execution of this activity required extensive hours of planning and testing on the part of the instructor and computing center personnel. Once all CMC activities were in place, the workload of the instructor continued to increase due to the extensive amount of electronic dialogue. However, scheduling of tests, assignments, and appointments with students was made much easier because of CMC. On days in which there was extensive e-mail traffic, the instructor spent most of the day responding to the students' questions and requests. In the context of a liberal arts education, however, this effort is a justified and necessary consequence of the technology. Perhaps in the future, more of the students can seek technical assistance with their peers or other experts instead of relying solely on the instructor.

Discovery Learning and the Culture of Teaching: EDE 455

EDE 455, Issues in Elementary Teaching, examines problems confronting elementary teachers from philosophical, historical, and sociological perspectives. This course is taken in the first five weeks of the student teaching semester. Eleven seniors, all of them women, were enrolled in the course which met three hours a day, five days a week.

Modem communication was the method of CMC chosen for EDE 455. The Indiana Department of Education Access Network, IDEAnet, is a statewide telecommunications system designed specifically for K-12 teachers, administrators, and classrooms. At this point, IDEAnet is strictly a dial-up service with a toll-free phone number to sites outside of the Indianapolis calling area. Users can log on to the system, receive and send electronic mail

to others around the state, participate in online chats with other educators, or read and send bulletins regarding any of over 70 discussion groups and conferences. Furthermore, IDEAnet supports a database of local school reports from which teachers and administrators can add or download information. IDEAnet even offers an employment referral service. All of these features of IDEAnet were explored during EDE 455.

The overall goal of our CMC activities was to enhance the pre-professional development of future teachers. The purpose of using IDEAnet was for the students to discover the power and relevance of telecommunications for themselves. IDEAnet embodies the interdependence between higher education institutions, the Indiana Department of Education, and the teachers, students, and administrators of all of the K-12 schools in the state. Since the conferences and information available on IDEAnet are current, the students learn to make use of the technology to benefit their careers.

Another function of using CMC was to strengthen the communication links between the individual student and professor. As the final course before their student teaching, it was hoped that EDE 455 would sufficiently prepare students to establish an IDEAnet station at their assigned school. The student teacher and his or her Franklin faculty advisor would then be able to continue a dialogue via IDEAnet.

Learning Activities

Two telecommunications stations were established for student use. One was located in the Curriculum Resource Center (CRC) of the College, a facility designed expressly as a resource and work room for future teachers. This facility had two Apple IIe computers and two Apple IIgs computers. IDEAnet was accessed from a single IIgs using a 1200 baud modem and Apple Access II terminal emulation software. The CRC is staffed during all business hours as well as some evenings and weekends, so the students could access IDEAnet at their convenience. Students also had limited access to their instructor's DOS-based 386 computer using Microsoft Windows Terminal software and a 9600 baud modem. With these two options, we were able to accomodate the technical glitches that arise in modem communications and provide access to two computing platforms.

The instructional objective for the class was to collect data on two demographically dichotomous schools or school systems. The class was divided into three groups (two groups of four students; one group of three students), each assigned a specific comparison: rural vs. urban schools; rich vs. poor schools; and large vs. small schools. The instructor selected the schools for the groups and provided training on IDEAnet. The students were evaluated on the thoroughness of their use of IDEAnet and other resources.

The premise behind this approach was that if we put the students into an environment rich with current and relevant data, the students could search through the databases, uncover the important facts, and induce from these relevant educational hypotheses.

Some of the activities with IDEAnet included:

Remote Databases:
- Searching strategies
- Accessing state-wide information
- Collection and analysis techniques of educational data
- Appreciation for accessibility and currency of on-line information

Asynchronous CMC:
- Dialogue with peers in other institutions
- Consulting with their advisor
- Online conferences
- Bulletin Board Services
- Modem communication technology
- Archiving of electronic mail and remotely accessed files

Observations

All of the eleven students attained complete proficiency with the IDEAnet system. Working in small groups or individually, the students had little difficulty in learning IDEAnet and constructing search strategies of its databases. They were all extremely motivated to use IDEAnet and their time on task extended well beyond class hours. Their patience with the system was very high. In fact, one student who worked in the CRC observed the students using IDEAnet and was motivated to log on himself! He, like the students enrolled in EDE 455, expressed a positive attitude about using IDEAnet and telecommunications in general.

All of the students were able to archive their messages and initiate online chats. The students, and indeed the instructor, were surprised at the friendliness and willingness of "strangers" on the network--administrators, teachers, researchers and other students--to participate in chats. In most cases, when the teacher or administrator on the network found out the person on the other end was a college education student, they were especially helpful with the IDEAnet system and in providing the would-be teacher with encouragement and assistance. Upon review of their dialogue, the instructor was pleased to discover high level conversations covering professional and educational issues. After these chats, the students would then continue correspondence via the IDEAnet mailbox.

In rare instances, a few EDE 455 students encountered negative discussions with elementary students on IDEAnet. This is one problem with an open-access network but our students were able to dismiss these experiences without loosing interest in the overall goals of CMC.

Most of all, working with IDEAnet gave students the opportunity to pursue meaningful educational research using real and current data. IDEAnet provided an easy to use front-end to official Indiana Department of Education databases. Using this information, students created portraits of schools and school systems. Students found this exercise to be more substantial and meaningful than textbook case studies. No reasonable alternative for doing this exists in a manner that is as easy and effective as IDEAnet.

Unfortunately, the continuation of these activities beyond EDE 455 have been disappointing. As the students began their student teaching experiences, their desire to continue using IDEAnet from their new site has met with resistance due to economic and philosophical reasons. Many administrators have been hesitant to make the necessary equipment available to student teachers. Of the eleven students, five are now located at a school that has extensive computing equipment, including telecommunications. IDEAnet, however, is not being promoted and is considered a "luxury". Three other students are assigned to sites with access to Prodigy, which means they have the necessary equipment for IDEAnet. The other three students are located at schools with limited computing hardware. The workloads and stress levels of student teachers is such that extensive changes must take place in order for telecommunications to be incorporated into their student teaching experiences.

CONCLUSION

Network communication was used in CMP 130 not only for demonstrating electronic mail, but also to integrate all of the course computing activities into a common network environment. Functional projects were designed to require extensive collaboration so as to necessitate CMC and common file access. We encountered a significant increase in the quantity of communication between the instructor and the students. Furthermore, incidental communication activities evolved which enhanced learner confidence with the environment.

In EDE 455, students learned to participate in on-line discussions and discovered for themselves the value of CMC. They also learned the technology of networking and how Indiana's dial-up educational network, IDEAnet, can let them reach into the participating classrooms of Indiana's K-12 schools. Students were able to pursue relevant educational research using current data.

Components of a telecommunications literacy

The International Society for Technology in Education recently made its recommendations to the National Council for Accreditation of Teacher Education (NCATE) for standard skills and competencies for teachers (Kinnaman, 1992). One recommendation included a knowledge of telecommunications and another specified that teachers be able to "use computer-based technologies to access information to enhance personal and professional productivity" (p. 9). To these general guidelines and from current trends in academic computing, we can add our experiences and observations to suggest a literacy for telecommunications and networking. Some recommended skills upon graduation include:

1. Awareness of the ethical and social courtesies of electronic communications (CMC)
2. Experience with modem communications and protocols
3. Competance with electronic mail functions
4. Experience with file transfer using utilities such as *Kermit*
5. Familiarity with online computing services
6. Knowledge of the Internet and the NREN
7. Practice with anonymous FTP and list servers
8. Competance with searching remote databases

This list is by no means exhaustive. Furthermore, any curriculum for education and other traditionally non-technical majors must consider these essential skills and not luxuries.

Telecommunications networks have major implications for the future of undergraduate instruction at even the small, liberal arts colleges. Until the past few years, however, this technology has not been a common component of most computer literacy paradigms. Within the next few years, it will be imparative that we provide the appropriate training in the areas of CMC, telecommunications and networking in general. Therefore, undergraduate curricula must include these activities in as many courses as possible.

Now that educators have the prospect of a national computing "highway" known as the National Research and Education Network (NREN), even small colleges will not be able to avoid the importance of telecommunications and CMC. At Franklin College, we want our students to be leaders in their chosen careers and to do this they must be proficient with this technology and its applications. The experiences gained as a result of these two courses has helped us instructors as much as the students meet these instructional goals.

REFERENCES

Bannon, L. J. (1986). Computer mediated communication. In D. A. Norman & S. W. Draper (Eds.), *User Centered System Design* (pp. 433-452). Hillsdale, NJ: Lawrence Erlbaum.

Blaschke, C. L. (1988). Networking in education. *Learning Tomorrow: Journal of Apple Education Advisory Council, 4,* 311-331.

Cohen, M. & Miyake, N. (1986). A worldwide intercultural network: Exploring electronic massaging for instruction. *Instructional Science, 26*(2), 257-273.

Kinnaman, D. E. (1992, January). What every teacher needs to know about computers. *Technology & Learning, 9.*

Levin, J., Rogers, A., Waugh, M. & Smith, K. (1989, May). Observations on electronic networks: Appropriate activities for learning. *The Computing Teacher,* pp. 17-21.

Levin, J. A., Kim, H. and Riel, M. (1990). Analyzing instructional interactions on electron message networks. In L. Harasim (Ed.), *Online Education: Perspectives on a New Environment* (pp. 185-213), Praeger, New York.

Mehan, H. (1989). Microcomputers in classrooms: Educational technology or social practice? *Anthropology & Education Quarterly, 20,* 4-21.

Quinn, C. N., Mehan, H., Levin, J. A. & Black, S. D. (1983). Real education in non-real time: The use of electronic message systems for instruction. *Instructional Science, 11,* 313-327.

Riel, M. M. and Levin, J .A. (1990). Building electronic communities: Success and failure in computer networking. *Instructional Science, 11,* 313-327.

Sproull, L., & Kiesler, S. (1986). Reducing social context cues: The case of electronic mail. *Management Science, 32,* 1492-1512.

Stuve, M. J. (1991). Exploring virtual classrooms: Network communication in a cross-cultural context. *Intelligent Tutoring Media, 2*(2), 51-62.

Teaching Elementary Matrix Operations Using Spreadsheets

David S. Tucker
Division of Mathematical Sciences
Midwestern State University
Wichita Falls, TX 76308

Abstract

Spreadsheets are gaining increasing popularity as a tool for teaching mathematical concepts from high school algebra to calculus,linear algebra, differential equations and even chaos theory, as well as business applications. This paper, a sequel to the paper Spreadsheets Macros as a Tool For Teachers [ASCUE, 1991], will take an in-depth look at creating spreadsheets to teach elementary row operations on matrices. A couple of different approaches are discussed and shown in detail. Some discussion is presented concerning the underlying philosophy of using spreadsheets in this, or other, mathematical applications.

Introduction

Electronic spreadsheet programs have been around since the early days of microcomputers. Invented originally for use in accounting, these programs have come to include many mathematical functions, graphing capabilities, and even a (macro) programming language. As early as 1985, some people began to recognize the value of spreadsheets as a tool for teaching mathematics, as well as accounting applications [1]. Each year the companies that produce spreadsheet programs add more capabilities and functions. One of the current trends is to use the power of the spreadsheet program for slide-show presentations, with the ability to merge text, graphics, and annotation.

In January, 1992, the Mathematical Association of America held its annual convention in Baltimore, Maryland. One session, chaired by Dr. Robert Smith of Miami University in Ohio, was centered on using spreadsheets in the teaching of mathematics. Thirty papers were presented on topics ranging from using spreadsheets to teach about functions in high school algebra to using spreadsheets in senior level college mathematics courses for topics like chaos theory, linear programming, or 2-D iterated maps.

Publishers are beginning to get on the bandwagon too. The first book on using spreadsheets to teach mathematics [1] is now being followed by other books such as [2] and [3]. As the versatility of spreadsheets for doing presentations increases, they become a natural medium for the classroom, perhaps even challenging graphics calculators [4].

At the 1991 ASCUE Summer Conference, the author presented a paper which discussed the basic spreadsheet platform as a programming language in which powerful applications can be written quite simply by people with little programming experience [5]. Experienced programmers can write applications using the macro programming language which are menu-driven and take advantage of all of the standard features of the spreadsheet program.

This paper, as a followup to [5], explores in some detail the implementation of matrix operations using a spreadsheet. It will look at several different possible implementations ranging from simple use of the /Data|Matrix operations provided by the spreadsheet, to an implementation written by the author which provides elementary row operations in a user-friendly, menu-driven format. In the process, several applications will be discussed, showing how this might be used in the classroom. For the purposes of this paper, the author has chosen to illustrate features of Lotus 1-2-3 and Quattro Pro -- perhaps the two most popular spreadsheet programs in use today. These same applications can be done in almost any spreadsheet program on the market today -- sometimes even more easily than illustrated here.

Supplied Matrix Operations in Lotus (and Quattro Pro)

Typical mathematical operations to be done with matrices are addition, subtraction, multiplication by a scalar, transpose, dot product of two vectors, matrix multiplication, and matrix inversion. All of these can be done with a spreadsheet, but perhaps not in the way one might expect. Both Lotus 1-2-3 and Quattro Pro provide a minimum of matrix operations as advanced mathematics functions. In Lotus they appear under the menu /Data|Matrix, and in Quattro Pro under the menu /Tools|Advanced Math. At first glance they seem to be quite limited because the only operations offered are matrix multiplication and matrix inversion. The reason for this is that the other operations can be done using formulas and operations on ranges.

The philosophy of the spreadsheet seems to be that the user will store a matrix in a range of cells. After all, a range of values is nothing more than a rectangular array of numbers -- the definition of a matrix. If the user wishes to do a binary operation like addition, then the two (conformable) matrix operands A and B are placed in two separate ranges, say A1..D3 and A5..D7, and then the formula +A1+A5 is placed in, say, cell A9. This cell is the upper left-hand corner of a range in which to store the resulting matrix, namely, the range A9..D11. The formula is then copied to all cells of the range (A9..D11), and the sum A+B of the matrices appears in that range of cells.

To take a difference of the two matrices in ranges A1..D3 and A5..D7, the formula +A1-A5 is placed in cell A9, and then copied thoughout the range A9..D11. For multiplication of the matrix in A1..D3 by the scalar 5, one puts the formula 5*A1 in the upper left-hand corner of the output range A9..D11 and then copies the formula to the rest of the range. The transpose of a matrix (i.e., make each row into a corresponding column) can be accomplished by using the command /Range|Transpose (/Edit|Transpose). This command prompts for the range in which the matrix is stored, and then prompts for the output range (which must be different).

For matrix multiplication of two matrices A and B, the /Data|Matrix (/Tools|Advanced Math) menu contains a multiply command which will prompt for the range of matrix A and the range of matrix B and a range for the output, and then will perform the operation, if it is legal. The matrix inversion command is also in this menu and will prompt for the range of the matrix and an output range and then perform the inversion. Using these tools, one might solve a system of three linear equations in three unknowns as the example below from [3:p194]. If we call the coefficient matrix of the system A, the column of constants B and the column of unknowns X, then the system

$$\begin{aligned} 2x+3y-z&=1\\ 3x+5y+2z&=8\\ x-2y-3z&=-1 \end{aligned}$$

can be written in the form AX = B, where

$$A = \begin{vmatrix} 2 & 3 & -1 \\ 3 & 5 & 2 \\ 1 & -2 & -3 \end{vmatrix} \quad X = \begin{vmatrix} x \\ y \\ z \end{vmatrix} \quad \text{and} \quad B = \begin{vmatrix} 1 \\ 8 \\ -1 \end{vmatrix}$$

The solution of this system can be found by evaluating the equation:

$$X = A^{-1} B$$

Using spreadsheets, we can do this as illustrated in the figures below:

	A	B	C	D	E	F	G
1	TITLE:		Three Simultaneous Equations				
2							
3	DESCRIPTION:		This worksheet demonstrates the use of an				
4			inverse matrix in solving a system of				
5			three simultaneous equations.				
6							
7	PARAMETERS:		A Matrix				B Matrix
8			2	3	-1		1
9			3	5	2		8
10			1	-2	-3		-1
11							
12	COMPUTATIONS:		A Inverse Matrix				X Matrix
13							
14							
15							

Figure 1

To implement this illustration, place the matrix A in the range C8..E10, the matrix B in range G8..G10. The INVERSE matrix will be in the range C13..E15, and the X matrix in the range G13..G15. Then do the following steps:

1. Select /Data|Matrix|Invert (/Tools|Advanced Math|Invert) and specify the range of A for the input matrix and the range of INVERSE for the output matrix.

2. Select /Data|Matrix|Multiply (/Tools|Advanced Math|Multiply) and specify INVERSE as the first matrix, B as the second matrix, and X for the output.

The inverse of the matrix A will appear in the range INVERSE, and the solution for the system of equations will appear in the matrix X, as illustrated in the next figure.

	A	B	C	D	E	F	G
1	TITLE:		Three Simultaneous Equations				
2							
3	DESCRIPTION:		This worksheet demonstrates the use of an				
4			inverse matrix in solving a system of				
5			three simultaneous equations.				
6							
7	PARAMETERS:		A Matrix				B Matrix
8			2	3	-1		1
9			3	5	2		8
10			1	-2	-3		-1
11							
12	COMPUTATIONS:		A Inverse Matrix				X Matrix
13			-0.5	0.5	0.5		3
14			0.5	-0.22727	-0.31818		-1
15			-0.5	0.318181	0.045454		2

Figure 2

This spreadsheet is an example of what is called a "template." It can be used to solve any consistent system of three equations in three unknowns by simply replacing the values in A and B and following the steps listed. If you need to solve a larger system, it is necessary to add more rows and columns and increase the size of the ranges accordingly. Also, if the matrix A is ill-conditioned, the Inverse operation provided with the spreadsheet might not be able to find the inverse.

Elementary row operations

One of the important concepts taught in any mathematics course involving matrix theory is that of elementary row operations on matrices. There are three row operations which are generally taught: The first is the operation of multiplying each element in a specified row by a nonzero constant (C*Ri); the second is interchanging any two rows (Ri <--> Rj); and the third is to replace a row by the sum of itself and a constant multiple of another row (Ri <-- Ri + C*Rj). Each of these operations can be performed on a given matrix by first performing the operation on an identity matrix (of appropriate size) and then left-multiplying the given matrix by the transformed identity matrix. This, of course, can be set up in a template form similar to the template illustrated above for the solution of three equations in three unknowns. As an illustration of this method, consider the following template:

	A	B	C	D	E	F	G	H	I	J	K
1	Elementary Row Operations On Matrices										
2											
3	Range names: E - for elementary matrix (A8..C10)										
4	A - for target matrix (E8..H10)										
5	O - for output matrix (E13..H15)										
6											
7	Elementary Matrix				Matrix A						
8	2	0	0		1	2	3	4			
9	0	1	0		5	6	7	8			
10	0	0	1		9	10	11	12			
11											
12					Output Matrix						
13					2	4	6	8			
14					5	6	7	8			
15					9	10	11	12			

Figure 3

In this example three ranges are created: range E for the elementary matrix, range A for the matrix A itself, range O for matrix which results from the product EA. This last matrix in the example above is the result of performing the elementary row operation 2*R1 (multiplying the first row by 2). The steps that should be taken are as follows:

1. In the range E (A8..C10) place a 3 X 3 identity matrix;

2. In the range A (E8..H10) place the matrix A;

3. Manually perform the row operation on the elementary matrix E (in this case that means changing the 1,1-position to 2, as illustrated above).

4. Select /Data|Matrix|Multiply (/Tools|Advanced Math|Multiply) and specify range E (A8..C10) for the first matrix, range A (E8..H10) for the second matrix, and range O (E13..H15) for the output.

The computer will then left-multiply the matrix A by the elementary matrix E, and place the result in O. We note that the matrix A does not change.

It is often the case that one wants to perform a series of elementary row operations on a matrix. This is done, for instance, in solving a system of three equations in three unknowns, or in finding the inverse of a matrix. We note that since the matrix A does not change in the above example, we must modify both E and A before performing the next elementary operation. We can modify A simply by using the /Copy (/Edit | Copy) command to copy the contents of range O to range A. Then we modify the elementary matrix to the one needed for the next operation and do the matrix multiplication again. It is worth noting that the ranges for the matrix multiplication will be the same, so that when going through the /Data | Matrix | Multiply (/Tools | Advanced Mat | Multiply) menu you can simply accept the default ranges shown for the matrix multiplication -- that is, you need only press <Enter> instead of pointing the required ranges.

It is an interesting "feature" of a spreadsheet that for the purposes of multiplication in this case, the range for the output can actually be the range of the right operand A. This observation leads to an obvious modification of the above example. By specifying the output range to be that of the matrix A, we need only modify the elementary matrix when we want to do a series of elementary row operations. For example, the spreadsheet might look like that illustrated below:

	A	B	C	D	E	F	G	H	I	J	K
1	Elementary Row Operations On Matrices										
2											
3	Range names: E - for elementary matrix (A8..C10)										
4	A - for matrix A (E8..H10)										
5	A - for output matrix (E8..H10)										
6											
7	Elementary Matrix				Matrix A						
8		2	0	0		1	2	3	4		
9		0	1	0		5	6	7	8		
10		0	0	1		9	10	11	12		
11											

Figure 4

Having set up the above spreadsheet to perform the operation 2*R1, after the multiplication operation we get the result:

	A	B	C	D	E	F	G	H	I	J	K
1	Elementary Row Operations On Matrices										
2											
3	Range names: E - for elementary matrix (A8..C10)										
4	A - for matrix A (E8..H10)										
5	A - for output matrix (E8..H10)										
6											
7	Elementary Matrix				Matrix A						
8		2	0	0		2	4	6	8		
9		0	1	0		5	6	7	8		
10		0	0	1		9	10	11	12		
11											

Figure 5

We see that the result is placed in the range of matrix A, and all we need to do is to modify the elementary matrix in E and multiply again. Thus a series of elementary row operations can be done quite rapidly.

Once again, to handle larger matrices, we need only increase the sizes of the ranges E and A. The problem we eventually run into is that we may get a matrix which is too big to fit on the screen (even the elementary matrix may become too large if we have a problem with, say, 30 rows and 40 columns -- not an unreasonable expectation in a linear programming problem.)

Menu-driven Solutions

In order to go beyond the simple use of templates, it is necessary to go into the macro command language which is supplied with modern spread- sheets. A discussion of some of the programming aspects of the spread- sheet is given in [5]. The matrix operation spreadsheet discussed there was designed to be as flexible as practical. The following goals were considered in the construction of this application:

- The user need not know how to use a spreadsheet
- The spreadsheet should accept a problem of any reasonable size
- The menus should be logical and convenient, and they should allow for a wide variety of needs and applications

A startup routine (\0) eliminates the need for the user to do any more than start the program and load the spreadsheet. Once the spreadsheet is up, the user is in a menu system which supplies all needed actions. From then on, the only instruction in the use of spreadsheets that needs to be done is that of how to select a menu option and how to enter values in the cells of the spreadsheet.

The first problem for the programmer is one of how to set up the spread- sheet to accommodate various sizes of matrices. To be able to do this, one must be able to calculate the cell address of the lower right-hand corner of the matrix (the upper left-hand corner will be fixed). This address is required in order to define a range for the matrix. The older versions of Lotus 1-2-3, such as version 2.01, do not have a way to calculate a column address conveniently, since the column address consists of either one or two letters. This oversight was corrected in Lotus 1-2-3 version 3 by adding the @COORD() function. This function will allow you to calculate a worksheet number (1 to 256), a column number (1 to 256), a row number (1 to 8192), and, after specifying a fourth parameter (which is a number from 1 to 8 and tells which of the three parts should be absolute, and which relative), the function will return a cell address. For instance,

@COORD(3+6,5*7,36-14,2) will return $I:AI$22

Note that the column address is relative, while the worksheet and row address are absolute. This comes from the choice of 2 for the fourth parameter. Version 3 allows multiple worksheets in a single file. They are denoted by A:, B:, etc. The nineth one would be I:.

If you are working with a version that does not have the @COORD function, or its equivalent, then there are several ways you might proceed.

First you might use the @CHAR function to create a one or two character string corresponding to the column address. For instance, if the upper left-hand corner of the matrix is to be put in cell B101, and the matrix is to be 5 X 8, then the row address of the lower right-hand corner is 100 + 5 = 105, and the column address of the cell is "I" = @CHAR(65+8). This method becomes a little more awkward, however, if the matrix is to be of dimension 29 X 30. Then the row address of the lower right-hand corner is easily computed as 100 + 29 = 129, but the column address consists of the two letters AE (recall that we are starting in column B, not column A). If we try @CHAR(65+30) we get "_", the underscore character. What we really need is

@CHAR(64+@INT(C,26)) concatenated with @CHAR(65+@MOD(C,26)) when the column size C is at least 26. In this case

@CHAR(64+@INT(30,26)) = @CHAR(64+1) = @CHAR(65) = "A",
and @CHAR(65+@MOD(30,26)) = @CHAR(65+4) = @CHAR(69) = "E"

A macro language routine do do this in Lotus 1-2-3 might be created as follows:

First we assume that two cells in the spreadsheet have been assigned range names NUMROWS and NUMCOLS, respectively. These are the cells which will hold the response of the user when asked how many rows and columns the matrix is to have.

The user is then asked to enter the number of rows and columns. If either number is zero, the program returns to the main menu. Note that the number is entered as a label instead of a number. This is because the {GETNUMBER} command has the unfortunate property that it will return an error if the entry is left blank. This would require the actual entry of the number 0 in order to go back to the main menu without completing the routine. By using a label, the user can change his mind simply by pressing <Enter> instead of entering a size for the matrix.

Through the use of @CHAR, @INT, and @MOD, the number of columns is turned into a string of one or two characters which represent the column address of the last column in the matrix. Since the row address is a number, we need only add the specified number to 100 to compute the desired row address. It is an interesting feature of spreadsheet macros -- what can be called "self-modifying programs" -- that we can store the results of these two calculation in cells further down the program listing. When control reaches those cells, the information individual to this particular run is now available as part of one of the program statements. This works because of the interpretive nature of the execution of a macro program.

The statements at lines 12 through 15 form the set of instructions to actually define a new range to go with the name MAT.

```
    K     L      M     N      O     P      Q      R
1   NUMROWS    35   NUMCOLS    40
2
3   ** MACRO TO SET UP INITIAL MATRIX **
4
5   {GETNUMBER "ENTER NUMBER OF ROWS: ",NUMROWS}
6   {IF @VALUE(+NUMROWS)=0}{RETURN}{LET NUMROW,@VALUE(+NUMROWS)}

7   {GETNUMBER "ENTER NUMBER OF COLUMNS: ",NUMCOLS}
8   {IF @VALUE(+NUMCOLS)=0}{RETURN}{LET NUMCOL,@VALUE(+NUMCOLS)}

9   {LET LASTCOL,@CHAR(64+@INT(+NUMCOL,26))&@CHAR(65+@MOD(+NUMCOL,26))}

10  {IF NUMCOL<=25}{LET LASTCOL,@CHAR(65+NUMCOL)}
11  {LET LASTROW,@STRING(+100+NUMROW,0)}
12  /RNCMAT~B101..
13  AO        <-- LASTCOL
14  135       <-- LASTROW
15  ~
```

Figure 6

12 /RNCMAT~B101.. is the instruction /Range|Name|Create together with the range name MAT (the ~ means <Enter>) and the listing of the first cell in the range -- B101 -- with the indicator to anchor the range (i.e., the ..)

13 AO is simply the computed designation for the last column, i.e., the value of LASTCOL

14 135 is simply the computed index of the last row, i.e. the value of LASTROW put into a string, since all macro commands are strings

15 ~ is the indicator to execute <Enter> to end the command

Note that the values in K13 and K14 go together to make up a single cell address. Both values must be string, but the spreadsheet program will automatically concatenate them together to form the cell address when this segment of the macro program is executed. The row designation is converted to a string using the @STRING() function.

When the program was originally written, a different method was chosen for computing the address of the last cell in the range. Instead of using string functions, a table lookup was used. This involved creating a table which contained values for the row and column designations of the last cell in every allowable size for the matrix, but it was felt that there were two advantages to using this method. First of all, it takes advantage of a table lookup function included in the spreadsheet. Since accounting applications often require table lookups for taxes, discounts, etc., the routines for this are quite fast. Secondly, the information, once put in the table, would be available for other routines in the application program. Table lookups involve specifying the value to look up in the table, a range name for the table, and an offset for information in the row where a match is found . Thus the program will search down the first column until it gets a match, and will then offset the specified number of columns to find the desired piece of information. The command might look like:

@VLOOKUP(+$ROWNUM,TABL,4)

In this example, the value stored in the cell with range name ROWNUM would have to be a string.

A third way to arrive at the last cell in the matrix would be to actually point out the range, using the fixed upper left-hand cell as a starting place, and the {RIGHT n} and {DOWN m} commands. This routine might appear as follows:

```
    K    L    M    N    O    P    Q    R
1   ** MACRO TO SET UP INITIAL MATRIX **
2
3  {GETLABEL "ENTER NUMBER OF ROWS: ",NUMROW}
4  {GETLABEL "ENTER NUMBER OF COLUMNS: ",NUMCOL}
5  {WINDOWSOFF}{PANELOFF}
6  /RNCMAT~B101..B101~
7  {GOTO}UPLEFT~/RNCMAT~.
8  {RIGHT @VALUE(+NUMCOL)-1}
9  {DOWN @VALUE(+NUMROW)-1}
10 ~{GOTO}B101~
11 {PANELON}{WINDOWSON}
12
```

Figure 7

One advantage of this method, is that, like the first method, the size of the matrix is limited only by the size of the spreadsheet. With the table method, the values in the table form a limit on the size of the matrix. Note that at line 6 we define the range MAT to consist of the one cell B101. This is done just in case for some reason the range MAT has been set with another upper left-hand corner prior to this initialization. Setting it to B101 here will release any previous setting. Lines 8 and 9 actually point out the new range. Lines 5 and 11 contain instructions to turn off the movement of the display and turn it back on again. This will prevent the screen from moving while the routine is being executed. This is especially important since the instructions of {RIGHT} and DOWN} are cursor movement instructions. Also, this will speed up the routine in the case of a large matrix which will not all fit on the screen at once. The fact that for large matrices the cursor has to travel through a lot of cells to point out the range, makes this method a little slower than the others. Actually, almost all of the routines in this application use the {WINDOWSOFF} and {PANELOFF} commands, but they will not be shown herein the interest of brevity.

Once the matrix is set up, there is the question of how to do the elementary row operations. We could hide an identity matrix in an unused portion of the spreadsheet, and use the matrix multiplication technique illustrated in the templates above, but that would make the size of the spreadsheet unreasonable for large applications. Instead, we choose to do the operations as they are defined in mathematics, using only the actual rows specified in the operation.

For instance, by using one or more rows hidden off the screen, say in rows 95 to 97, we can duplicate any of the values we desire. In this application, the visible region for the user is the portion of the spreadsheet starting at row 100 and going through larger numbered rows. The next three figures show the routines which do the three elementary row operations. They each assume that the initialization routine has defined a range TMPROW1, which is nothing more than row 95 of the spreadsheet, starting at column B and containing exactly as many columns as the matrix being operated upon. In the case of the first and third elementary row operations, a formula which performs the operation with the values from the designated rows is placed in the first cell of TMPROW1, and then copied to the other cells in the range. This produces the values which should be put in Ri. The contents of the cells in TMPROW1 are formulas, however, so we use the /Range|Value command to copy the values of the cells into Ri.

In the case of the interchange operation, we simply use TMPROW1 as a set of temporary variables for the standard three-statement interchange routine.

Also note that, in these figures, what is displayed here is the actual formula in each cell. When they are typed into the spreadsheet, commands like @STRING(100+ROW1) will display the value instead of the formula.

As discussed in [5], spreadsheet macro programs have a minimal amount of error handling. One example is the use of the function @ISERR in line 4 of the routine for multiplication of a row by a scalar. This is necessary, as describe above, since we stored the user's response as a numeric value with the {GETNUMBER} command, and if the user either just pressed <Enter> or entered a label, the value stored in ROW would be ERR, and the program would die. The solution here is simply to abort the routine and go back to the menu which called it. This same idea could have been used in the initialization routine (Figure 6). However, because there is need to process the number of rows and the number of columns as strings in some instances and as numbers in others, it really would not save much work.

To clarify this point, when the value in a cell like SCALAR is used to create part of a command (see line 9 of Figure 8), a string is required because all commands go into cells as labels. When the value in ROW is used to create a numeric value by adding 100 to it, as in line 11 of Figure 8, it must be a numeric value. Use of the @STRING() and the @VALUE() functions allows you to make the change easily.

90

```
    M       N       O       P       Q       R       S
 1  ** ROUTINE TO MULTIPLY A ROW BY A SCALAR (C*Ri) **
 2
 3  {GETNUMBER "ENTER THE NUMBER OF THE ROW TO MULTIPLY: ",ROW}
 4  {IF @ISERR(ROW)}{LET ROW,0}{IF +ROW<=0}{RETURN}
 5  {IF +ROW>NUMROW}{RETURN}
 6  {GETNUMBER "ENTER THE SCALAR TO USE: ",SCALAR}
 7  {IF @ISERR(SCALAR)}{LET SCALAR,1}{IF +SCALAR=0}{LET SCALAR,1}
 8  {GOTO}B95~
 9  @STRING(+$SCALAR,0)
 10 *B
 11 @STRING(100+$ROW,0)
 12 ~
 13 /C~TMPROW1~
 14 /RVTMPROW1~B
 15 @STRING(100+$ROW,0)
 16 ~
```

Figure 8

One might wonder about the use of the command {CALC} in row 29 of Figure 9 and in row 51 of Figure 10. This command forces a recalculation of the spreadsheet. Although this adds a bit to the time, it is sometimes necessary to force this recalculation when a series of row operations is being done. Experience shows that occasionally the routine will get ahead of itself and use values from ROW1 or COL1, for instance, before the cell has been updated with the most recent entry. {CALC} forces that update, thus assuring that the proper row or column is used.

```
    M       N       O       P       Q       R       S
 21 ** ROUTINE TO INTERCHANGE TWO ROWS (Ri <--> Rj) **
 22
 23 {GETNUMBER "ENTER THE NUMBER OF THE FIRST ROW: ",ROW1}
 24 {IF @ISERR(ROW1)}{RETURN}{IF +ROW1<=0}{RETURN}
 25 {IF +ROW1>NUMROW}{RETURN}
 26 {GETNUMBER "ENTER THE NUMBER OF THE SECOND ROW: ",ROW2}
 26 {IF @ISERR(ROW2)}{RETURN}{IF +ROW2<=0}{RETURN}
 28 {IF +ROW2>NUMROW}{RETURN}
 29 {CALC}{GOTO}B
 30 @STRING(100+ROW1,0)
 31 ~/C
 32 {RIGHT +NUMCOL-1}~TMPROW1~
 33 {GOTO}B
 34 @STRING(100+ROW2,0)
 35 ~/C
 36 {RIGHT +NUMCOL-1}~B
 37 @STRING(100+ROW1,0)
 38 ~/CTMPROW1~B
 39 @STRING(100+ROW2,0)
 40 ~
```

Figure 9

Although these routines are the basis for the elementary row operations, many utility routines have to be added in order to make the program user friendly. The use of a menu-driven system means that the cursor movement keys serve the purpose of selecting menu items. Thus it is necessary to create separate routines to allow the user to scan the matrix -- especially if the matrix is too big to fit on the screen. An example of such a routine is given in [5]. On the other hand, the nature of any spreadsheet is that the cursor moves to a cell and either puts a new value in it, or displays or uses the value already there. Even when the screen and panel are turned off, the cursor is logically moving in the spreadsheet as the commands are executed. It is then necessary to have a routine which will keep track of the starting position and return the cursor to that position so that there is a minimal amount of disruption for the user. Other routines to print or save the spreadsheet, change the format or width of the cells, exit from the spreadsheet, enter the values into the matrix, or simply return to the upper left-hand corner of a large spreadsheet, must be added.

```
    M       N       O       P       Q       R       S
41  ** ROUTINE TO PERFORM  (Ri <-- Ri+C*Rj) **
42
43 {GETNUMBER "ENTER THE NUMBER OF THE ROW TO CHANGE: ",ROW1}
44 {IF @ISERR(ROW1)}{RETURN}{IF +ROW1<=0}{RETURN}
45 {IF +ROW1>NUMROW}{RETURN}
46 {GETNUMBER "ENTER THE NUMBER OF THE OTHER ROW: ",ROW2}
46 {IF @ISERR(ROW2)}{RETURN}{IF +ROW2<=0}{RETURN}
48 {IF +ROW2>NUMROW}{RETURN}
49 {GETNUMBER "ENTER THE SCALAR: ",SCALAR}
50 {IF @ISERR(SCALAR)}{LET SCALAR,0}
51 {CALC}{GOTO}TMPROW1~
52 +B
53 @STRING(100+ROW1,0)
54 +B
55 @STRING(100+ROW2,0)
56 *
57 @STRING(SCALAR,12)
58 ~/C~TMPROW1~
59 /RVTMPROW1~B
60 @STRING(100+ROW1,0)
61 ~
```

Figure 10

Conclusion

Matrix operations and elementary row operations find a wide variety of uses in mathematics. The basic philosophy by which this menu-driven elementary row operation spreadsheet was built is that the spreadsheet is to take the place of the calculator in performing the arithmetic operations. This assures more uniformity in the results the students turn in, but it also means that the student must learn the concept being taught -- namely what sequence of row operations to perform, and how to interpret the results.

This program has been used quite successfully in an operations research course to teach how to solve systems of equations, find the inverse of a matrix, and do the pivots of the simplex algorithm. Because the program does nothing more than help the student to perform series of elementary row operations, the program is not limited as to where and when it may be used.

It is possible to actually set up tutorial applications with a template or spreadsheet macro program, but the real power of spreadsheets lies in their simplicity and versatility. Students can be encouraged to explore. They can actually design their own spreadsheets as projects, or the teacher can use the spreadsheet in classroom demonstrations. Spreadsheets can serve nicely to find "quick and dirty" solutions, or they can be used on a regular basis. Any way you look at it, spreadsheet programs are a powerful tool for the teaching of mathematics.

References

1. Arganbright, Deane E. Mathematical Applications of Electronic Spreadsheets, McGraw-Hill, 1985.

2. Spero, Samuel W. The Electronic Spreadsheet and Elementary Calculus: Graphing and Numerical Methods, Harper-Collins Publishers Inc., 1991

3. Etter, D.M. Lotus 1-2-3 Release 2.3: A Software Tool for Engineers, Addison-Wesley Publishing Company, Inc., 1992

4. Arganbright, Deane E. "Teaching With Spreadsheets: Advanced Applications and Creative Graphics", a paper presented at the Joint Mathematics Meetings of AMS and MAA, Baltimore, MD, January 1992

5. Tucker, David S. "Spreadsheet Macros as a Tool for Teachers", Proceedings of the 1991 ASCUE Summer Conference, June 1991

Case Studies in Systems Analysis and Design Course and Senior Project Course

Jill Gray
Mathematics & Computer Science Department
North Georgia College
Dahlonga, GA 30597

ABSTRACT:

At North Georgia College, we have just implemented a new CPSC480 - Senior Project course in our CpSc curriculum. This course is a second quarter continuation of the CPSC361 - Systems Analysis and Design. In the Senior Project course we plan to complete the design and implementation phases of a project developed in the immediately preceding analysis and design course.

We decided on a teaching strategy of allowing the Systems Analysis and Design students to each choose a project on their own. The projects pertain to clubs, activities and problems with which the students have personal experience. The results have been particularly interesting and encouraging. The students are actively competing to accomplish a well developed and complete case study, in order that theirs be chosen for the Senior Project.

The freedom to pick their own Systems Analysis and Design project is a subject of controversy among educators. The arguments against this teaching strategy include the students lack of business experience and the enormous work load put on the instructor to track each case and student progress. The payoff could be student enthusiasm and involvement which is the basis of a good learning experience.

INTRODUCTION:

The Senior Project course, CPSC480, is a new addition to our curriculum at North Georgia College. It is a one quarter follow-up course to the one quarter Systems Analysis and Design course, CPSC361. We decided to add the Senior Project course in order to have enough time in which to have students complete a project - from project proposal through implementation and a first round of maintenance. We also wanted to get our curriculum more in line with the ACM Small College Task Force recommendations for curriculum content.

We decided on a teaching strategy of allowing the Systems Analysis and Design students to each choose a project on their own. The projects proposed and chosen for case studies pertain to clubs, activities on campus and off, part-time jobs, and problems with which the students have personal experience. Successful projects have included systems for membership and dues tracking in a campus fraternity, gift giving, local contractor/builder registration, Bass Fishing Tournament information, and campus newspaper cost accounting. These projects, though limited in scope and size, provide a familiar and manageable project for an individual student. The results have been remarkable.

RATIONALE:

Our students have very little or no business or accounting experience. Case studies presented in text books include applications such as payroll, inventory, workers compensation insurance, and class registration systems. These case studies, even the class registration systems that is a process the students are familiar with, include business problems that are beyond their experience. Consequently, as the instructor, I seem to need to spend more time explaining common business and accounting functions than on the problem solving and systems analysis tools for the course work.

Using individual projects as the framework for the Systems Analysis and Design course requires that the students work on their own and create each and every phase of a completely designed project. We select

a text that follows a formalized sequence of analysis and design phases, presenting the theory and philosophy, methods and tools needed to complete that phase in the predefined sequence. The students get practice doing each phase of the project and using each of the problem solving skills and tools presented, as well as the experience of maintaining continuity and completeness throughout the project.

Our goal is to complete the analysis and most of the design phases of the project in the first quarter course. From there, we go on to complete the design, implement the system and do the first round of maintenance in the second quarter course. Usually, by the end of the first quarter, the projects include a Problem Definition document, current system documentation in the form of Data Flow Diagrams and systems charts, an Entity Relationship Diagram, and proposed system documentation in the form of Data Flow Diagrams, decision trees and a Project Dictionary. In the second quarter, a project is chosen from those created during the first quarter. That project is then taken through the completion of the design and implementation phases, including design of files, outputs, inputs and programs, implementation of these designs, followed by the first round of maintenance activities.

The Senior Project course, the second quarter, requires that the students work as a project team, under the direction of a project leader. This gives them the experience of working together in the kind of team environment used in most 'real world' situations. The project leader is the student whose Systems Analysis and Design case from the previous quarter was chosen to be implemented. The choice is made by the instructor based on completeness and correctness of the analysis and design phases assigned, as well as interest and applicability. The students must each contribute and cooperate in order for the implementation to be successful. The maintenance activities again give the students a taste of the 'real world' in that projects are often not completed before modifications to the original specifications are requested by the users. This activity also serves as a recap of the entire systems analysis, design and implementation process, as each phase in the process is revisited in order to assess, design and implement the requested modifications.

COURSE ACTIVITIES:
SYSTEMS ANALYSIS AND DESIGN:

The course work follows the typical ordering and constraints of a one quarter analysis and design class. Topics include the systems development life cycle, problem solving techniques, methodologies and tools. The case study activities include a feasibility study, process modeling with Physical Data Flow Diagram, data modeling with Entity Relationship Diagrams, process modeling with Logical Data Flow Diagrams, and defining logical data and information requirements in a Project Dictionary.

A text book case is chosen by the instructor to be presented in class as an example of each of the activities listed above. This case is developed in class by the instructor and documented consistently through each phase. As the development progresses, small parts of this sample case are assigned to the students as practice exercises, such as a physical data flow diagram for a particular process. When this is done, it is particularly important to spend class time checking for consistency and completeness in the entire set of diagrams produced. This process gives the students a working model from which to pattern their individual case study projects.

The text book case study is an important part of classroom activities, but not the only component. The theory and philosophy of systems analysis are at least equally important and deserve equal class time. Lectures are reinforced with the exercises and study question/problems assigned from the text.

Individual case studies are done primarily outside of class time. Each phase is assigned a due date some reasonable length of time after the activities associated with that phase have been completed in lecture, the text and in the sample case study. At appropriate points in the individual project development, class time is taken to discuss and share knowledge and information about the activities carried out by individual students. An example of such an activity would be informal student presentations made while developing system alternatives in the feasibility study phase. Discussions at this point encourage the students to share their knowledge of hardware, software and other types of manual and mechanized options. At the end of the quarter students are

required to make a twenty minute presentation of their project, including feasibility, cost justification and user requirements for the proposed system.

SENIOR PROJECT:

This course completes the design activities, then focuses on implementation and maintenance of a complete system. Topics include the systems development life cycle, conventional and data base file design, designing and prototyping outputs and controls, designing and prototyping inputs and controls, designing and prototyping on-line terminal dialogues, designing methods and procedures, structured programming, testing, and project management techniques and tools.

The text book case developed in the previous course is continued as an example of activities that the project team is expected to accomplish. As each topic is presented in lecture, the lecture is followed with a fully developed output using the text book case. The instructor goes through the entire process in class. The students are familiar with the case and can easily relate to the activity. Actual implementation can be restricted to a small portion of this case.

Again, the sample case study is not the only component of classroom activities. Lecture and class time must also be devoted to the theory and philosophy of design, implementation and maintenance, as well as techniques and tools. Text book study questions and problems are assigned to the students to reinforce this aspect of course work.

Activities of the project team are done primarily outside of class time, except for short planning sessions and instructor update activities which are done during class time. The planning sessions are run by the student project team leader with the assistance of the instructor as necessary. I have found that very little, if any, assistance is needed, except attending these meetings. The instructor update sessions are intended to keep the instructor informed as to project progress and to give the students a chance to practice their presentation skills. These sessions emulate the project reporting sessions usually given by a team leader to a superior within the department and/or a user or group of users. The team leader is encouraged to assign portions of the reporting to members of the team. When the project has been implemented, the team is responsible for a formal presentation on the capabilities of the system and an informal training session for the user. After the maintenance phase, the team is again responsible for both the formal presentation of capabilities and user training. The team leader is also required to maintain a project schedule, project team assignment log and project diary.

GRADING:

In the Systems Analysis and Design course, CPSC361, the grading is completely individual. Sixty percent of the students' grade is based on the project, an equal portion for each assigned phase in the analysis and design sequence. Thirty percent of the grade is assigned for the presentation of the project made by the student to the class. The other 10% is reserved for class participation and class assignments made on the sample case and textbook study/essay questions.

Grading in the Senior Project course is entirely on participation as a member of the project team and completion of assigned portions of the project. The team leader is responsible for project completion and coordination with the instructor.

COURSE ADMINISTRATION:
SYSTEMS ANALYSIS AND DESIGN:

As was previously discussed, the primary motivating factor in letting the students develop their own case studies is their lack of general business and accounting background. But, this decision also represents the whole gambit of administrative problems with the course.

The case studies chosen by the students must be approved by the instructor and followed by the instructor through all phases of development. This requires that the instructor be familiar with each case and conversant with all aspects of the general business and accounting practices encountered. In order to evaluate a case study, the instructor must devote significant time and energy to each proposal and selected case study project. Activities included are interviewing and approving each user for the proposed case study, developing an understanding of the problem to be solved through student documentation and user interaction, checking for completeness and accuracy throughout the project development by carefully grading student work and maintaining contact with the user, and also check the student work for completeness and correctness in terms of generally accepted business and accounting practices.

These activities are no small accomplishment. If you have move than 10 students in your class, you may be tempted to have the students work together in groups of 2 or 3. I have found this to be a reasonable solution, but not nearly as effective for individual student learning.

SENIOR PROJECT:

Course administration is a breeze compared to the first quarter course. The instructor administers the course primarily through the student team leader, planning meetings and reporting sessions as previously described. Deadlines must be approved for specific phases of design completion, implementation, testing, presentations and user training. Some project teams are nearly self-propelled, but others occasionally need prodding.

CONCLUSION:

You will find this an interesting and challenging teaching experience, because of the enthusiasm, creativity and dedication of your students. The diversity of projects, users and potential solutions will expand your knowledge and practical experience more than you would ever expect.

The students write comments on their course evaluations each quarter. As always, some students are critical of the whole concept, writing such comments as "I don't see what this course has to do with computers." But by far the majority of the students are pleased and enthusiastic about the experience. Most write comments such as "I finally know why I had to learn about indexed file structures," or "The project brings everything I've learned over the past four years together so that it all makes sense." That is exactly what I hope to accomplish as a Computer Science instructor.

Teaching Data Communications

Louis M. Janelle, Jr.
The University of Akron - Wayne College
Orrville, Ohio 44667

This presentation describes methods for teaching an introductory semester course in data communications at the two-year college level. The course addresses the use of modems, bulletin boards, facsimile devices, scanners, and local area networks.

The course is divided into two parts: asynchronous data communications and local area networks. Topics under the heading of asynchronous data communications include the following: purchasing modems, choosing communications software, using electronic funds transfer, contacting bulletin boards, downloading files, experimenting with RS232 serial communications, using electronic mail, transmitting documents with internal and external facsimile devices, and scanning documents and separating graphics from text.

The segment on local area networks covers a variety of topics: logging in on a local area network, setting passwords, creating user and group accounts, modifying login scripts, creating network menus, and mastering Novell Netware utilities. Lab activities, which form the core of the course, serve to clarify concepts and to motivate students to experiment with these communications tools.

The presenter will share the lab manual developed for the course. A diskette containing lab activities and references will be distributed at the session.

Teaching Programming with Closed Laboratories

David E. Maharry
Department of Mathematics and Computer Science
Wabash College
Crawfordsville, IN
MaharryD@Wabash.Bitnet

For two semesters Wabash College has used closed laboratories as an integral part of CS1, the Introduction to Programming course. The underlying philosophy of these labs is that students learn a language by reading and using well-written code. The students then test their understanding of the language by modifying this code, adding new sections, procedures and functions which are similar, or are extensions, of existing code. Students learn to build reusable code by using a set of labs in which ideas or code developed in one lab are reused in later labs and code developed in the labs are found useful in their individual programming assignments. Lectures and homework assignments focus on methods of problem solving and algorithm design rather than on the syntax of a particular language.

The paper will describe examples of these sets of labs and both some of the successes and some of the problems that have been discovered with this method.

Wabash College, a liberal-arts college of 850 men in Crawfordsville, and DePauw University, a liberal-arts college of 2200 men and women, shared a grant from the Pew Charitable Trust Foundation which was given to support collaborative work in Science education. As part of this grant, Professor Carl Singer of DePauw and Professor David Maharry of Wabash worked together to redesign the introductory computer science course offered by each school. This collaboration began during the summer of 1991 and continues. At Wabash, CS1, Introduction to Programming, is offered each semester with an enrollment of about 25 students per semester. It is a half-course, meaning approximately one and one-half to two semester hours of credit. This course focuses entirely on the introduction to programming. The roughly equivalent course at DePauw is a one credit course, or about four semester hours and includes other topics besides programming, such as some history of computing, computer architecture, and applications of computers. This paper will focus on the course that was developed and tested at Wabash College.

CS1, Introduction to Programming, has been offered at Wabash, in one form or another, for over 25 years. Until the summer of 1991, it was offered in a format which had two lectures per week. The lectures, each of 50 minute duration, provided examples of programs, syntax of the language, and the development of algorithms. Some weekly homework exercises were assigned but the main assignments were a set of programming assignments, usually four of five during the semester. These assignments required the student to write a program, working individually. The assignments were progressively more difficult, beginning with a simple program that would ask the user to enter some data, the program would perform a calculation and print the results. Later assignments would introduce the ideas of procedures, loops, arrays, and data files. These assignments were written, entered, tested and run outside of class and then turned in to be graded.

Perceived Problems with the CS1 Course

A number of difficulties with this method of teaching programming were perceived. 1) The students did not see the importance of writing software that was reusable. They were not encouraged to go back to earlier assignments or earlier programs to begin building the next program. 2) The students did not have the opportunity to work with well-written, readable, software to see what that idea meant. They were not asked to modify working software to function in a slightly different way or to extend software to accomplish more things. 3) Students would discover problems in their programs or they would find programming ideas that they did not understand late at night as they struggled to complete an assignment just before it was due and in a place where the instructor was not available to help them. 4) The early programs were very simple and not very exciting.

This seems more important now then it did a few years ago before young people were as familiar with the powerful video games of recent years. 5) The instructor was not as aware of the stumbling blocks that the students ran into as he would be if he were there at the time of the stumble.

A New Direction

All of these perceived difficulties, coupled with the boredom of the instructor that sets in after teaching a course for over twenty straight semesters, propelled us in a new direction. During the summer of 1991, the course was redesigned by developing a series of closed laboratory activities around which the course was centered. This new organization has been used for two semesters now at Wabash and for one semester at DePauw. It is still under revision but the main structure is well in place.

The Use of Class Time

The new course outline involves one lecture per week, with a duration of one and one-quarter hours. Thus there are fourteen lectures per semester. These lectures describe the ideas of procedures, parameters, the modular style, loops, selection, and algorithm design. Algorithms such as those for finding the maximum in a list, sorting items, and searching for an item in a list are discussed. An emphasis is placed on the method of how to approach problem solving and design. Different data types such as strings and arrays are discussed and used in the algorithms that are developed. Very little time is spent on the particular syntax of a programming language.

Using Closed Labs Each Week

Weekly scheduled laboratories are held. The labs are two hours in length and are scheduled in a micro-computer equipped classroom where each student works individually at a computer. In each of the laboratories, the student begins by retrieving a complete, well-written program. The activities each week involve reading the lab program, then running the program, and observing how it works. The student is then asked to modify the program. In most labs this begins by merely change part of the program. Then the student is asked to add new features to it which are similar to features all ready in the program. In each of these steps they are encouraged to see a pattern in the parts of the program so that they can abstract out of that what they want to use in their modified version.

The student is asked to use the debugger as a part of several of the labs. The student is encouraged to see that he is able to watch the execution of the program, observing how the values stored in memory are used and modified. Having some experience in using the debugger in these labs, the student is able to see the usefulness of this tool as he works on later labs or on his programming assignments.

Each lab has several goals. One is to show a student a programming construct, such as loops or procedures, or to show a student an algorithm that was discussed in class such as simulation, sorting or searching. The second goal of these labs is to introduce the student to the syntax of the language, Turbo Pascal. The earlier labs demonstrate the programming constructs such as procedure definition and invocation, variable declaration, while loops, if-then-else statements, etc. and ask the student to modify them and then develop sections of programs themselves. Thus the student is faced with the task of determining the correct syntax in the midst of a problem to solve. The instructor is there for guidance and suggestions.

Labs and Programming Assignments

The nine structured labs form three sequences of three labs each. Each sequence of three labs are designed around one of the three programming assignments. To encourage students to do their own original work on out-of-lab programming assignments, it is important to have different programming assignments each semester. To make it possible to use the lab activities each semester but have different programming assignments, it was decided that three general application areas would have to be chosen around which the labs could be centered and these would form the general areas for the programming assignments. The three areas that were chosen were graphics, simulation and text processing. Thus each semester the first assignment asks the user to write a program that produces some graphic effect such as a part of a video game. Each semester the assignment is a different variation around this general theme. The second assignment is a computer simulation of some type and the third involves some kind of file manipulation and text processing.

With this in mind the first three lab activities involve working with programs that display and work with graphic images. These labs develop a set of procedures, functions and ideas that the student is able to use to develop his own program to solve the first programming assignment which is due at the end of the fourth week of the semester. In addition, these first three labs introduce the student to the ideas and syntax of declaring variables, assigning them values, defining and using procedures while passing parameters by value and by address, designing and using loops and boolean expressions, and selection statements. All of these programming constructs are introduced in the context of a problem to solve. They are introduced as part of a working program which the student is asked to modify and extend. Each lab builds on the labs before with the student being asked to cut definitions and procedures from the program that he developed in an earlier lab and paste them into the new lab that he is writing. From the very beginning of the semester the student is encouraged to see that some procedures or functions are general enough to be reused in later work.

To allow the student to develop programs, even the very first one, which are visually interesting or are able to solve interesting problems, a library of procedures, functions and objects has been developed which a student is encouraged to use in his labs and programs. For the first set of labs, a figure object has been defined which has methods to set the size and color of a generic figure, retrieve the size and color of the figure, to move it to a new position on the screen and to retrieve its current position. It has no shape or ability to be seen. The first lab shows the student how to define a derived object, which has the shape of a square, by defining a Display method. As part of the lab he is asked to use this definition as a model and define a new figure of some other shape, a triangle for example. In just a few minutes in the first lab, he finds that he can write a procedure and use it to move and display his own figure, changing its size and color. Students are encouraged to see that all figures have certain articles in common, they move, they have color, they have some measure of size, etc. and they can build on this to develop their own programs.

The second lab uses the figure definitions that the student developed in the first lab but extends the work to include moving the figure smoothly across the screen. This necessitates repeating a sequence of statements, displaying the figure, hiding it, moving it to a new location, displaying and hiding it again, and so forth. A loop is needed and so becomes a part of this new program.

The third lab again builds on the ideas and procedures of the first two. It has a procedure asking the user to enter a starting position for the moving figure. The procedure contains a selection statement which checks whether the data that was entered was valid, was the position on the screen, and assigns valid values if the ones entered were not valid. The student is asked to write a similar procedure to request the speed for the figure.

The lab also contains code to detect if a key has been pressed and if it is a 'G' key, increase the size of the figure, if it is a 'S' key shrink the figure, and if it is a 'Q' key then the program should halt. The student is asked to modify the code to allow the user to change the speed of the figure by pressing other keys.

Finally the lab asks the student to consider what should be done if the figure reaches a boundary of the screen. If the figure reaches an edge do one thing otherwise do something else. The need to use a selection statement is seen.

This past semester the first programming assignment asked the student to develop a program that would be part of a video game, moving a figure down the screen in some random direction, while the user can move a paddle back and forth across the bottom of the screen trying to intercept the falling object. Much of what they need for this program they have already developed and tested in the labs. They see that they can use this earlier code, cutting and pasting, modifying when needed.

In each lab students found that they could extend what was being asked. For example, rather than just defining a triangle, similar to the square that they were given, students were seen to define diamonds, stars, even an airplane. They were playing with their code the first week of CS1! These new, individualistic figures showed up then a few weeks later in their video games. Many of the programs turned in for the assignment included such additions as figures that changed size or color every time they hit an edge of the screen, or increased their speed according to the laws of gravity as they dropped down the screen.

The fourth week there was no organized lab activity. The lab was open and the instructor was there but this was a time for the students to work on completing their assignment. The instructor was there to help with problems, suggest directions to go to find solutions, get excited when the program came together. This idea of an open, or programming, lab time scheduled the week before a programming assignment was due was well used and appreciated by the students.

The fourth, fifth and sixth labs focused on the idea of computer simulation. Programs that simulated flipping coins and rolling a die were run and modified. A queue object was introduced with a lab which allowed the user to enqueue and dequeue items, watching the queue grow and shrink. In one lab arrays were needed to keep track of the number of times each face of the die appeared. A histogram object was introduced to allow the program to display the number of times each face was rolled on the die. During these labs the student was asked to use the built-in debugger to trace the steps of the program, watching the values of some of the variables.

The second programming assignment asked the student to simulate a gas station, with cars arriving randomly to get service. The output included a histogram of the percent of time that there were no cars in line, the time with one car, the time with two cars, etc.

The seventh and eighth labs focused on working with files and text. Labs included opening files and reading them, displaying their content and finding words in a line of text. The last lab asked the student to write a procedure to sort a list of numbers using one of the sorting methods that had been discussed in class. Using a clock object that was part of the library, they were asked to find the time it took to sort lists of different sizes and then to draw a graph of this relationship using a plotting object that was also part of the library.

The third programming assignment asked them to write a cross-reference program, reading a program file, and finding all of the words that were not reserved words, sorting these into alphabetic order and storing these identifiers in a text file. As before, this involved pulling together procedures and functions that they had developed over the previous three labs.

Observations and Evaluations

As the second semester of this course comes to an end, there are a number of observations that can be made, some positive and some negative. On the positive side, it seems apparent that the student interest is greater in the new course. They get excited at being able to develop what they see as 'useful' software right away.

They quickly catch on to the idea of reusing software, both software given to them in the labs and software that they develop. Some algorithms appear often enough that they seem to expect it to be in every program and wonder if they missed something if it is not. The algorithm to find the largest element in an array is an example of this. It seems to magically show up in four of the labs and two of the assignments.

The students see well-written software, well-commented and with good style, right from the start. In fact, they are including this code in their own code and it rubs off. They recognize with their own eyes that well-written code is easier to understand.

The instructor is more aware of student problems right when they occur, right in the midst of the struggle, not after the student has sledge-hammered a solution at midnight while working under a deadline that the lab will close and the assignment is due at 8AM the next morning.

The programming assignments seem to be completed on time more easily since the students are beginning to develop the code week by week. When they come to the lab that fourth week, just before the assignment is due, their program is usually pretty well in hand. Students also seem to begin the programming assignments with a clearer idea of how to proceed. They are not starting with a blank slate.

Some Concerns

On the negative side, it was found that the labs take a great deal of time to develop and even after they have been developed they continue to need to evolve, although the time that takes is considerably less.

It also takes time to supervise the labs. At Wabash, each lab could have no more than 20 students so it was necessary to run two labs each week, taking four hours of supervision. This replaced 1 1/4 hours of lecture.

Assignments must change from semester to semester but need to be tied to the labs. This requires the labs and assignments be generic yet specific. This may be difficult over several semesters.

It is difficult to know how to evaluate the lab part of a student's work. With the labs so much a part of the assignments, it was felt that the students would recognize the importance of the labs, so a grade in lab was based solely on attendance, not on the work done. This seemed to be a good assumption as the students attended, worked hard, and accomplished a great deal in lab. In two semesters, no one has left early or skipped a lab. However it was felt that they often came to lab with out doing the work that was expected before hand. Thus we now give a short quiz at the start of the lab. It is intended to be very easy if the student has read the assigned chapters in the book and the lab handout that is given out in the class before the lab day. This seems to have helped some but a problem still exists.

One concern is that the labs give them a good deal of structure to bring to their programming assignments. This is important at the beginning. However in the current version of CS1 there is no assignment where they are asked to supply their own overall design, working from scratch. This past semester in CS2, Advanced Programming, in which all of the students have learned their programming in the lab-based CS1, the students seemed to move easily to the much larger programming projects, although the first asssignment in CS2 was assigned in parts, providing some structure. At the time of this writing, the evaluations of the CS2 course, which ask specifically for the students to comment on their background from CS1, have not been completed.

Clearly there are a number of concerns that this style of course has raised. However, the overall feeling is one of excitement. Both the students and the instructor seem to come away with this sense of enthusiasm. Wabash and DePauw are committed to continue to develop this course and its laboratories.

File Structures Using C

Dr. Patricia T. Montague
Dr. James R. Van Doren
Department of Mathematical Sciences
Metropolitan State College of Denver
Denver, CO 80217-33672

ABSTRACT

ANSI C has defined a standard I/O library to handle input and output operations which includes functions to process files from a low-level point of view, i.e., as a stream of bytes. The C program using a file determines what bytes are to read or written.

Most books on C include at most a brief treatment of the fundamental file operations. This paper discusses these and gives some example programs including a hex dump program and a co-sequential time line merge program which illustrates some encapsulation principles for file processing. The material included was selected to provide a person with some C experience the background needed to teach or use file structures in C.

The advantage of using ANSI standard C in the study of file structures is that it allows one to consider a file as a stream of bytes. The standard C I/O library provides file functions such as **fopen**, **fread**, **fwrite**, **fseek** in **<stdio.h>** and these are discussed in this paper. In order to see what is actually stored in a binary file, one may need to use a hex dump. A hex dump program is included along with examples using command line arguments for the input of the names of the files. The paper ends with a discussion of time line algebra problems and how they can be objectively implemented using ANSI standard C.

Variations of the two presented programs can provide an appropriate basis for teaching several important ideas in a file structures course. These include, but are not limited to, binary files, co-sequential processing and encapsulation of a file object.

STREAMS

All input and output operations in C take place thru calls to functions in the C standard library. The input/output model is based on an abstraction which is called a stream. This provides a constant interface which is independent of the actual device being accessed. The actual device is called a file. The file can be what one normally considers a file, that is, a disk file, or it can be an external device such as a printer or a terminal. There are two types of streams -- text and binary.

A text stream is a sequence of characters composed into lines. (This is actually allowed and not required by ANSI C.) Each line contains zero or more characters and is terminated by the newline character. The newline character is optional on the last line. ANSI C requires that lines be allowed to have up to 254 characters. A one-to-one correspondence between the characters in the text stream and the characters in its external representation is not required. As long as characters are added, deleted, or altered in such a way as to be transparent to the user of the text stream, input/output can be modified to conform to the host environment. An example of this is mapping the newline character, **'\n'**, to carriage return and line feed.

A binary stream is a sequence of unprocessed bytes. There is a one-to-one correspondence between the bytes in the stream and those on the external device. It is possible that there is an implementation dependent number of null bytes appended to a file. This might be done to fill out a sector of a disk.

Streams can be unbuffered, line buffered, or fully buffered. In an unbuffered stream, characters are transmitted immediately to or from the external representation. In a line buffered stream, this transmission takes place when a newline character is encountered. In a fully buffered stream input transmission takes place when the buffer is empty and output takes place when the buffer is full. The function **fflush** when applied to an output file causes the unwritten contents of an output buffer to be transmitted to its external representation.

FILES IN ANSI C

Every stream is associated with a structure of type **FILE**. (**FILE** is a type that is declared in **<stdio.h>** and is implementation dependent.) This association is created when the file is opened and a pointer to the structure, referred to as the file pointer, is created and returned by **fopen**. The parts of the structure **FILE** are the file-position indicator, error indicator, end-of-file indicator, and a pointer to the internal buffer associated with the file.

The file-position indicator gives the position in the file which is being written to or read. In a binary file, the file-position indicator is incremented as each character is processed. In a text file, the manipulation is defined by the implementation since it depends on how the end of a line is represented. When an error occurs the error indicator is set to a value that encodes the type of error. When an attempt is made to read past the end of the file the end-of-file indicator is set. Some implementations require that these error indicators be reset before any further processing of the file -- this is done by **clearerr**. In **<stdio.h>**, the macro **EOF** is usually defined to be **-1** and the macro **NULL**, the null pointer constant, is defined to be one of **0** or **0L** or **(void*)0**.

When a file is closed, it is automatically flushed (i.e., any unwritten buffer contents in memory are transferred to the external representation). All files are closed automatically when the program terminates normally; they are not closed when the program crashes. The function **fclose** releases the file control block associated with the file so that it is available for reuse.

When a C program is started three text files are automatically opened. These are standard input (**stdin**), standard output (**stdout**) and standard error (**stderr**). The file pointers **stdin**, **stdout**, and **stderr** are constants. If not redirected, standard input is the keyboard and standard output and standard error are the screen. Note that normal output and error messages are in separate files and that one can have standard output redirected to a file while the standard error messages are displayed on the screen.

UNIX-LIKE FILE SYSTEM

Most implementations of ANSI C have a UNIX-like file system which is not defined by the ANSI standard. The major I/O functions for this system are **read**, **write**, **open**, **close**, and **lseek**. This is sometimes referred to as the unformatted, or the unbuffered, file system. In this system the programmer has to provide and maintain all disk buffers. Many implementations of C do not allow one to use the ANSI file functions and the UNIX-like functions in the same program.

In this system, a file descriptor of type int is used. The prototypes for the specific I/O functions are as follows:

```
int open(const char * filename, int mode)
int creat(const char * filename, int mode)
```

The parameter **mode** is a macro defined in **<fcntl.h>**. Standard modes are **O_RDONLY** for read only, **O_WRONLY** for write only, and **O_RDWR** for read and write. Other modes such as text and binary may be supplied with a compiler. If the value returned by open is -1, then the file can not be opened; otherwise a positive integer value is returned which is the file descriptor for the opened file. Some compilers do not allow one to use **open** on a file that does not exist. The function **creat** is used to create a new file.

BEST COPY AVAILABLE

int close(int fd)

The parameter **fd** must be a valid file descriptor which has been obtained thru a call to **open** or **creat**. The value **-1** is returned if the file can not be closed. The value zero is returned if **close** is successful. The **close** function releases the file descriptor so that it can be used by another file and flushes the internal disk buffers of the operating system. There is usually a limit as to the number of files that can be open simultaneously.

int read(int fd, void * buf, unsigned size)

Each time a call to **read** is executed, **size** number of characters are read from the file specified by **fd** to the memory pointed to by **buf**. If the read is successful the number of characters actually read is returned. If it is unsuccessful, it returns zero at the physical end of file and **-1** if errors occurred.

int write(int fd, const void * buf, unsigned size)

Each time a call to **write** is executed **size** number of characters are written to the file specified by **fd**. The characters are taken from the allocated portion of memory, or the variable, pointed to by **buf**. The number of bytes written is returned by a successful call to write. Generally **EOF** is returned by an unsuccessful call.

long lseek(int fd, long offset, int origin)

The parameter **offset** is usually a long integer. The **origin** may be one of the macros defined in **<io.h>**. These are **SEEK_SET** for the beginning of the file, **SEEK_CUR** for the current position and **SEEK_END** from the end of the file. **lseek** returns the current file position as measured in bytes from the start of the file. If **lseek** is not successful, then **-1L** is returned.

ANSI C FILE FUNCTIONS

The prototypes for the ANSI C file functions are given next along with comments about them. These are divided into groups of related functions with the open/close and status discussed first, then input/output, buffering, and finally random access.

FILE * fopen (const char * filename, const char * mode)

The parameter **filename** is a pointer to a string of characters that form a valid filename (this may include a path specification and the total length may not exceed **FILENAME_MAX** which is defined in **<stdio.h>**). The parameter **mode** is specified as follows:

"r"	open an existing text file for reading
"w"	create a text file for writing an existing file with that filename will be destroyed
"a"	append to an existing text file will create the file if it doesn't exist
"rb"	open an existing binary file for reading
"wb"	create a binary file for writing an existing file with that filename will be destroyed
"ab"	append to an existing binary file will create the file if it doesn't exist
"r+"	open an existing text file for read/write
"w+"	create a text file for read/write an existing file with that filename will be destroyed
"a+"	append or create a text file for read/write

"rb+" open an existing binary file for read/write (also "r+b")
will create the file if it does not exist
"wb+" create a binary file for read/write (also "w+b")
an existing file with that filename will be destroyed
"ab+" append a binary file for read/write (also "a+b")
will create the file if it does not exist

The mode of "a" sets the file-position indicator to the end of the file. This may be actually beyond the last character written to the file if null character padding has been added to the file. With the mode of "a" all writes are at the end of the file. The modes of "r" and "w" set the file-position indicator to the beginning of the file.

The ANSI standard specifies that at least eight files may be open at a time. The number that can be opened is specified by **FOPEN_MAX** in **<stdio.h>**. If the **fopen** is successful, the file pointer is returned; if not **NULL** is returned.

int freopen(const char * filename, const char * mode, FILE * fp)

Opens a file and associates it with the file pointer **fp** which is already in use, closing the original file. This can be used to change the access mode of an existing file.

int fflush(FILE * fp)

Flushes the buffer associated with the output or update file pointed to by **fp**. It returns zero if successful; **EOF** otherwise. If **fp** is **NULL**, then all files are flushed. If **stdout** is buffered, then a prompt that doesn't end in '**\n**' can be written out by using **fflush.**

int fclose (FILE * fp)

If **fclose** is successful, the value zero is returned; if not, **EOF** is returned. A failure is generally caused by the disk being full or the disk not being there. The function **fclose** frees the file control block associated with the stream so that it can be reused and it flushes any data remaining in the internal disk buffers. The function **exit** calls **fclose** for open files.

int feof(FILE * fp)

This function tests, but does not clear, the end-of-file indicator for **fp**. It returns zero if and only if the end-of-file indicator has not been set. This function is needed because the binary data read may be the same as the **EOF** mark.

int ferror(FILE * fp)

This function tests the error indicator for **fp**. It returns a nonzero value if and only if an error has occurred. The functions **clearerr** or **rewind** can be used to clear the error. Note: not all implementations have **ferror** return the error status.

int clearerr(FILE * fp)

Explicitly clears the error indicator and the end-of-file indicator for **fp**.

void perror(const * char)

This writes to **stderr** the string pointed to by **char**, then a colon, and then an implementation dependent message, corresponding to the current value of **errno**. If **char** is **NULL**, then no prefix is attached.

ANSI C FILE FUNCTIONS FOR INPUT AND OUTPUT

int fscanf(FILE * fp, const char * control_string, ...)
int fprintf(FILE * fp, const char * control_string, ...)

These correspond to **scanf** and **printf** except that input/output is to the file pointed to by **fp**.

int putc(int ch, FILE * fp)
int fputc(int ch, FILE * fp)
int putchar(int ch) is equivalent to **putc(int ch, stdout)**

The parameter **ch** is converted to an unsigned character (by using only its low-order byte) and then output to the file pointed to by **fp**. Usually **putc** is a macro and it appears to preserve compatibility with earlier versions of C. If the function is successful, **ch** is returned and the file-position indicator is advanced; otherwise, it returns **EOF** and the error indicator is set. Since the usual value for **EOF** is **-1** and C does not require that the type **char** hold a signed quantity, these functions return an integer instead of a character.

int getc(FILE * fp)
int fgetc(FILE * fp)
int getchar(void) is equivalent to **getc(stdin)**

These are equivalent functions that input the character pointed to by the file-position indicator. The function **getc** is usually a macro which preserves compatibility with earlier versions of C. If successful, the unsigned character read is converted to an integer (the high-order byte is zero) and then returned and the file-position indicator is advanced. **EOF** is returned and the end-of-file indicator is set when the file-position indicator is past the end of the file. If a read error occurs, then **EOF** is returned and the error indicator for the file is set. This differs from Pascal in which the function eof returns true when the next character to be read in the file is the end-of-file marker.

int fputs(const char * str, FILE * fp)
int puts(const char * str)

The function **fputs** writes a null-terminated character string to the file pointed to by **fp**. The null character is not written. It returns **EOF** if an error occurs; otherwise a nonnegative value. The function **puts** writes the string to **stdout** and the newline character is written in place of the null character.

char * fgets(char * str, int length, FILE * fp)
char * gets(char * str)

The function **fgets** reads a character string into the array pointed to by **str**. It reads from the specified file until the newline character is read or **length-1** characters have been read, or the end of file has been reached. If the newline character has been read, then it is stored in the string. Both functions append a null character to the end of the string. If an error occurs or no characters were stored, both return **NULL**; otherwise, they return **str**. The function **gets** reads from **stdin** until it encounters the newline character, which it does not store, or the end of the file. Note that using **gets** instead of **fgets** can allow the input of a line which is longer than the storage pointed to by **str**.

The functions **fread** and **fwrite** can read/write data types that are longer than one byte. They both advance the file-position indicator accordingly. The type **size_t** is an unsigned integral type that can not be longer than unsigned long. It is defined in **<stddef.h>** and in **<stdlib.h>** and is the type of the value returned by **sizeof**. For both functions, the file-position indicator has an indeterminate value if an error occurs.

size_t fread (void * buf, size_t num, size_t count, FILE * fp)

The parameter **buf** is a pointer to the array which will receive what is read and **num** is the number of elements of size **count** to be read. The function **fread** returns the number of items read. This value will be less than count if the end of file is reached or if there is an error. If either **num** or **count** is zero, then **fread** is successful.

size_t fwrite (void * buf, size_t num, size_t count, FILE * fp)

The parameter **buf** is a pointer to the array which is to be written and **num** is the number of elements of size **count** to be written. The function **fwrite** returns the number of items written. This value will be less than **count** only if there is an error.

ANSI C FUNCTIONS FOR FILE BUFFERS

A buffer is automatically established for a file when **fopen** or **freopen** opens it. This buffer is **BUFSIZ** bytes -- **BUFSIZ** is a macro in **<stdio.h>** and must be at least 256.

void setbuf(FILE *fp, char * buf)

This function changes the buffer from the default to **buf**. The buffer will be **BUFSIZ** bytes. If **setbuf** is called with **buf** as **NULL**, then the file will be unbuffered. The function should be called after the file has been opened, but before any reading or writing.

void setvbuf(FILE * fp, char * buf, int mode, size_t size)

This allows the buffer size to be specified by the parameter **size** and the **mode** of buffering to be set to **_IOFBF** for fully buffered, **_IOLBF** for line buffered, and **_IONBF** for unbuffered. (These macros are defined in **<stdio.h>**.) The value returned is zero if successful; a nonzero number if an invalid parameter is given for **size** or **mode**, or something went wrong.

ANSI C FUNCTIONS FOR RANDOM ACCESS

If a file is opened for random access using the mode **"+"** along with **"r"** or **"w"**, then the file-position indicator is initialized at the beginning of the file. As each character is read, or written, the file-position indicator is incremented accordingly. (Access mode **"a"** allows writing to be done only at the end of a file.)

int fseek (FILE * fp, long offset, int origin)

The function **fseek** sets the file-position indicator for the file pointed to by **fp**. The pointer **fp** must have been returned by a call to **fopen**. The parameter **offset** is the (signed) number of bytes from the origin which is to become the new file position and **origin** is one of the following macros defined in **<stdio.h>**:

SEEK_SET	beginning of the file
SEEK_CUR	current file position
SEEK_END	end of the file -- not required by the ANSI standard for a file opened in binary mode

The function **fseek** returns zero when successful and a nonzero value if an error occurs. It does not clear the error indicator for **fp**; it clears only the end-of-file indicator. In some implementations seeking past the end of a file, or before the beginning of a file, will not result in an error condition until a read or write is attempted at that position.

Note that an input operation may never directly follow an output operation, or vice versa; the program must call **fseek**,
or **fflush**, **rewind**, or **fsetpos** between the two operations.

long ftell(FILE * fp)

Returns the value (as a long integer) of the file-position indicator as an offset from the beginning of the file; -1L if an error. If the value returned is **-1L**, then the value of **errno** can be checked to see if an error actually occurred. Note: **ftell** in a text file returns an implementation-defined number. In Unix-style environments **ftell** returns the number of characters from the beginning of the file to the current position; in MS-DOS environments a line is terminated by a carriage return/newline pair and ftell counts each of these.

void rewind (FILE * fp)

Resets the file-position indicator to the beginning of the file. This is the same action as **(void) fseek(fp, 0L, SEEK_SET)**. The function **rewind** clears the error indicator for **fp**.

For large files -- files that are 4 gigabytes or larger, **fsetpos** and **fgetpos** should be used in place of **fseek** and **ftell**. These use **fpos_t** which is an implementation dependent type that is defined in **<stdio.h>**. Both of these return zero if successful; if not, a non-zero value is returned and **errno** is set to an implementation-defined error number.

int fsetpos(FILE * fp, const fpos_t * pos)

The function **fsetpos** resets the file-position indicator. The value of the parameter **pos** must one that has been returned by an earlier call to **fgetpos**.

int fgetpos(FILE * fp, const fpos_t * pos)

Copies the current value of the file-position indicator to the storage pointed to by **pos**.

HEX DUMP PROGRAM

The following program creates a file which is the hex dump of the file chosen by the user. It may be especially important in teaching a "creative" file structures course for students to have such a tool available. Such a tool can be used to reinforce the differences between text files and binary files.

The program is done in the modern style of returning a value to the operating system (using zero as a return value indicating successful completion). This is the purpose of using **return 0** at the end of the program. The function **exit** is used if the file to dump can not be opened -- the value returned is a **1**. One advantage of using **exit** is that this automatically calls the function **atexit** which can be used to specify functions to be executed upon termination of the program. Furthermore, **exit** provides a graceful termination of the program and can be called in main or in a function.

```
#include <stdio.h>
#include <ctype.h>
int main(void)
{  const char hex_code[] = "0123456789ABCDEF";
   FILE * filein, * fileout;
   char filenamein[15], filenameout[15];
   int count = 0;    long int total_count = 0;
   unsigned char chunk[16];    char hex_data[32];
   int k, i1, i2, i;
   printf("\nEnter the name of the file you wish to process:\n");
   gets(filenamein);
   printf("\nEnter the name of the output file for the dump:\n");
   gets(filenameout);
   fileout = fopen(filenameout, "w");
   filein  = fopen(filenamein, "rb");
   if (filein == NULL)
     { printf("file opening error -- program stopped\n");
       exit (1);  }
   while ((count = fread(chunk, 1, 16, filein)) > 0)
     {  for (i1 = 0, i2 = 0; i1 < count; i1++, i2 += 2)
          { hex_data[i2] = hex_code[chunk[i1]/16];
            hex_data[i2 + 1] = hex_code[chunk[i1]%16];   }
        for (i1 = 2*count; i1 < 32; i1++)
          hex_data[i1] = ' ';              /* blank fill */
        fprintf(fileout, "%04X", total_count);
        fprintf(fileout, " - %04X:", total_count + 15);
      for (i = 0; i < 32; i++)
        {  if (i % 4 == 0) putc(' ', fileout);
           putc(hex_data[i], fileout);  }
      putc(' ', fileout);    putc(' ', fileout);
      for (k = 0; k < count; k++)
         if (isprint(chunk[k])) putc(chunk[k], fileout);
                        else  putc('.', fileout);
      putc('\n', fileout);
        total_count += count;  }
   fclose(fileout);
   return 0;  }
```

The first four and the last four lines of the hex dump of the source code above are below. The computer used was a 486 microcomputer. Note that each line ends in the pair of hex numbers **0D0A** -- these are unprintable control codes, carriage return and line feed, and appear in the ASCII part on the right as two periods.

```
0000 - 000F: 2369 6E63 6C75 6465 203C 7374 6469 6F2E  #include <stdio.
0010 - 001F: 683E 0D0A 2369 6E63 6C75 6465 203C 6374  h>..#include <ct
0020 - 002F: 7970 652E 683E 0D0A 696E 7420 6D61 696E  ype.h>..int main
0030 - 003F: 2876 6F69 6429 0D0A 7B20 2063 6F6E 7374  (void)..{  const
  ...
0590 - 059F: 6F75 6E74 202B 3D20 636F 756E 743B 2020  ount += count;
05A0 - 05AF: 7D0D 0A20 2020 6663 6C6F 7365 2866 696C  }..   fclose(fil
05B0 - 05BF: 656F 7574 293B 0D0A 2020 2072 6574 7572  eout);..   retur
05C0 - 05CF: 6E20 303B 2020 7D                        n 0;  }
```

Changing the mode of the input file to "r" changes the pair of hex numbers 0D0A to 0A as is illustrated by the first and last four lines of the resulting hex dump. This illustrates a difference between binary files and text files where "filtering" may be applied by the underlying operating system, MS-DOS in this case.

```
0000 - 000F: 2369 6E63 6C75 6465 203C 7374 6469 6F2E  #include <stdio.
0010 - 001F: 683E 0A23 696E 636C 7564 6520 3C63 7479  h>.#include <cty
0020 - 002F: 7065 2E68 3E0A 696E 7420 6D61 696E 2876  pe.h>.int main(v
0030 - 003F: 6F69 6429 0A7B 2020 636F 6E73 7420 6368  oid).{  const ch
  ...
0570 - 057F: 6E74 202B 3D20 636F 756E 743B 2020 7D0A  nt += count;  }.
0580 - 058F: 2020 2066 636C 6F73 6528 6669 6C65 6F75     fclose(fileou
0590 - 059F: 7429 3B0A 2020 2072 6574 7572 6E20 303B  t);.   return 0;
05A0 - 05AF: 2020 7D                                  }
```

Since a text file is being used for the hex dump examples, the dumps do not illustrate the reversing of the bytes of the values of numbers which occurs with some computers (such as an IBM PC or a VAX). The decimal number 36 stored as two ASCII characters would be displayed in hex as **3336**. The decimal number 36 is represented in one byte in binary as 00100100 which would be **0024** in hex using two bytes in order and **2400** in reverse order. The ASCII form of **24** is **'$'** and of **00** is **'\0'** which is not printable. The resulting ASCII form for the pair is **.$** if the bytes were stored in order and **$.** if they were stored in reverse order.

For assignment purposes, a student may be given a hex dump program like the one above and then be required to produce a variation such as hex dumping selected portions of a file on command.

COMMAND LINE ARGUMENTS

The hex dump program above asks the user to input the name of the file to be dumped and the name of the file for output. C permits this information to be conveyed using command line arguments and a program can be set up so that if the arguments are not supplied, then the user is asked for it. Several different examples are included below. The last example is in Turbo C++.

```
main(int argc, char * argv[])
{  ...
   unsigned char fileanamein[15];
   unsigned char filenameout[15]
   if (argc < 2)                    /* no file names given */
      { printf("Name of file to dump: ");
        gets(filenamein)   }
    else strcpy(filenamein, argv[1]);
   if (argc == 3)
        strcpy(filenameout, argv[2])
    else
      { printf("Name of file for output of the hex dump; ");
        gets(filenameout);  }
    ...
```

Another way to do this is:

```
if (argc >= 2)
     (void) strcpy(filenamein, argv[1]);
 else
   { (void) fprintf(stderr, "what file to dump");
     (void) gets(filename);    }
```

One should recall that both **strcpy** and **fprintf** return integers. The result is type cast above to **void** by the **(void)** in order to avoid an error when checking the program using **lint**. Checking a program with **lint** is routinely done in some commercial applications of C.

A less friendly way would be to stop execution of the program if the two files names are not supplied on the command line. This is many times done in the following way:

```
if (argc != 2)
   { printf ("USAGE: %s filenamein filenameout \n\n", argv[0]);
     return 1;    }
```

The following sets up a program so that the default files are standard input and standard output:

```
FILE * filein  = stdin;    FILE * fileout = stdout;
```

The following Turbo C++ code gives a final example of the handling of command line arguments. This code assumes that the only the name of the input file is specified as a command line argument.

```
#include <fstream.h>
#include <string.h>
main(int argc, char*argv[])
{  ifstream ifile;
  ...
  if (argc <= 1)
     { cout << "\n\nUsage: hexdump <filename>\n\n";
       exit(1);   }
  ifile.open(argv[1]);
  if (!ifile)
     { cerr << "\n\nFile ";
       cerr.write(argv[1], strlen(argv[1]));
       cerr << " not found!\n";
       exit(1);    }
```

TIME LINE ALGEBRA

The purpose of this section is to explain the concept of time line algebra prior to presenting a non-trivial co-sequential merge program implemented in the form of separate source files. The term "time line algebra" is a term used to describe a collection of processes associated with the manipulation of lists of time records. These occur in certain types of scheduling processes where it is necessary to keep track of lists of time intervals during which a resource is available for allocation and also lists of time intervals which represent request for allocation of a resource. Merging and intersecting pairs of lists, which correspond to set union and set intersection, are typical algebraic processes on a time line. However, the time intervals represent a continuum of time and therefore algorithms for these operations need to reflect this fact.

A time interval is a record containing a start time and an end time or a duration. Other data may also be associated with a time record. Normally the time values are positive integers representing the amount of time since some base time (midnight, beginning of year, decade or century) in appropriate units of time. For algorithmic purposes the lists of time records are normally ordered on start time. Normally a "point interval" (start time and end time the same) is not allowed. Two time intervals are called juxtaposed if the start time of one record is the same as the end time of the previous record. Depending on the issue of "item specific" versus "pooled" resource, juxtaposed time intervals may or may not be appropriate. It may be appropriate to coalesce juxtaposed time intervals into one time interval. Time line management is also subject to the classical storage management issues of external fragmentation and internal fragmentation, because of the variable length time intervals -- this is outside the scope of this paper, however.

In some cases time intervals represent "windows" of time in which scheduling of a resource is requested. In such a case a time record will have at least three time fields - start, end, and duration.

If an activity such as sending data from a satellite to a ground station requests scheduled use of the ground station during specified lists of time windows with a given duration, then it is appropriate to "intersect" the time window list with the ground station availability list to find time intervals which can be used to satisfy the request. If scheduled use of a ground station by a satellite is removed from the schedule, then it is appropriate to merge the scheduled list of time intervals with the available list of time intervals. In practice significant complexities arise in the scheduling or rescheduling process to construct a "good" schedule for a limited resource and because there are many types of resources and ways to request scheduling of those resources.

In the satellite case issues of Aquisition of Signal - AOS - or line of sight, due to orbits and the rotation of the earth, become nontrivial issues in terms of constructing and managing lists of time intervals. In some cases it may be appropriate to "fragment" a request and satisfy it with scheduling multiple time intervals across available times of a resource such as a ground station. In some cases multiple resources need to be scheduled in a particular pattern for some activities. The complexities associated with such "activity" scheduling can be very significant. Having available robust algorithms which will function properly in a variety of circumstances, including empty input lists, may be an important ingredient in solving "the scheduling problem" which may exist.

Whether or not it is appropriate to permit overlapping time intervals across two lists is dependent on the circumstances. In the example below and in the program example in the next section, this type of overlap is permitted.

Below two lists of time intervals in the form of lists of ordered pairs of start time and end time are subjected to both a merge process and an intersection process to come up with resulting lists of time intervals. Algorithms for these processes can be based on co-sequential processing models such as the Van Doren model (Folk and Zoellick [7]) provided that the lists are ordered on one of the time fields, usually the start time.

List 1: (1,3), (5,7), (10,15), (16,17), (18,20), (23,29), (30,33)

List 2: (1,4), (7,16), (17,20), (20,25)

Result of merging: (1,4), (5,29), (30,33)

Result of intersecting: (1,3), (10,15), (18,20), (23,25)

Co-sequential processing algorithms for these types of processes can provide in a file structures course alternative and stimulating examples of the wide applicability of co-sequential processing.

SEPARATE SOURCE FILES

In this section two principal uses of separate source files are elaborated. A fairly complete example of the notions discussed herein is presented in the form of an implemented solution to a time line merging problem. This example can be used to reinforce important ideas of "encapsulation" associated with ADT's and object-oriented programming.

For further reference the terms "utility library" and "encapsulated object" are used to distinguish between the two principal uses of source files addressed in this paper. The principal difference between these two uses of source files has to do with the data, namely where the data are declared and how they are accessed. For utility libraries the data are normally declared and controlled by the invoking software and are passed to the utility functions by the standard parameter mechanism of C. Results are returned either by virtue of "call by reference" parameters (pointer access) or by function return value.

Commonly used utility functions associated with a given application, or a related group of applications, can be implemented with a distinct source file and be made available for all related applications. In this sense, an object file resulting from compiling a source file of utility functions serves a purpose similar to standard function libraries. Indeed, most implementations of C include utility software for the purpose of creating and maintaining user libraries (which may arise from many source files) or rely on standard operating system commands (i.e.; UNIX) for the same purpose.

For an encapsulated object the data are declared and controlled by the invoked software. The parameter mechanism and function return mechanism are used for delivering results as needed. The actual data associated with an object can be made "private" so as to prevent inadvertent access. Only publicly known functions can manipulate such data. Such encapsulation, as this has come to be known, can carefully associate object data and operations on those data in a controlled manner, thus decreasing the likelihood of uncontrolled and/or unintended access to the data, particularly critical control data. If need be, say for reasons of execution speed, some of an object's data may be made public to provide "public" access without sustaining the overhead of function invocation and return. Also some operations on an object's data may be made private so they may only be used by other publicly known functions in support of "approved" public operations. The insulation of this approach is useful in building "software parts" whose usefulness may exceed that of conventional libraries because both the data and the operations on the data are implemented in the object.

For either of these two principal uses of separate and distinct source files, the use of "header files" containing function prototypes for their respective source files is an appropiate ANSI standard method for providing "included" information about the usage of the functions contained in the respective source files.

But, alas, there thus become two related, but physically distinct, files for each utility library or encapsulated object. Therefore there exists the danger that such files may become inconsistent, resulting in potentially disastrous consequences which may be very difficult to track down. Fortunately there is a way to protect against such inconsistencies, namely by including the relevant header file in its corresponding source file! In this way an ANSI standard C compiler will be caused to verify the function prototypes against the actual definitions of functions in much the same manner that such a compiler will check function usage against prototypes, but with much greater rigor. The greater rigor is due to exact type checking between a prototype and its definition if they occur in the same file being compiled.

The two related files, a C source file and its corresponding header file, collectively correspond to the notion of a Turbo Pascal "unit".

The implementation of a utility library is quite straight forward. One codes the functions in the library in a standard manner, but using few or no global declarations of data. Relevant data are passed by parameters. The corresponding header file is abstracted from the function definitions, included in the source file itself for verification reasons mentioned above, and included in source files referencing any function in the utility library.

The implementation of an encapsulated object requires the use of global declarations for data and associated use of the "static" key word. This key word prevents a thusly declared data item from being known outside its source file; that is to say, its visibility is restricted. The fact of being global within the source file permits access to all functions forward of such a declaration. In effect, the static key word so used amounts to a declaration of "private." (The use of the static key word in a local context has a different meaning. In this context the scope is local, but the storage for the declared item is static storage rather than the default "automatic" storage class.) Global data not restricted by the static key word have an external storage class attribute and are, in effect, "public" data by which any source file may gain access by using an "extern" declaration.

A static declarative attribute may also be given to a function definition which similarly restricts its visibility so it is known forward of its definition, but not outside the source file in which it is declared.

A complete programming example is given below. The specific time line algebra problem is a time line merge implementation in which two input files of time records of nonoverlapping time intervals ordered on start time are co-sequentially processed to produce an output file of nonoverlapping time interval records. Rigorous sequence checking is applied to all input records and output records to prevent inadvertent errors in logic from propagating. Because empty lists (files) of time interval records commonly occur in time line algebra problems, rigorous attention is paid to the possibility of one or both input files being empty and thus also the possibility of an empty output file.

In contradistinction to typical co-sequential merge processes the number of output records may be dramatically smaller than the total number of input records, because time intervals across files may coalesce to form larger time intervals.

The implementation is presented in the form of two objects, one utility library, and a main program. In addition there are four header files, one each for the objects, one for the utility library, and one containing relevant type definitions for time interval records.

The two objects comprise an input file object and an output file object. The purpose of the main program is to co-sequentially process the input so as to pass this input to the output object in the requisite ordered manner. The rigor of sequence checking, end of file handling, and empty file handling is embedded in the objects. Clarity of purpose and simplicity of implementation of the main function is enhanced by the use of these objects. (A time line intersection program would use a very similar main program, an identical input object, but with a different output object, because internal control information and operational logic differ for time interval intersection.)

The two input files are implemented as one object rather than two for reasons of privately shared data for end of file control and because the input files are identical in form. The utility library comprises two functions used by both objects, namely file opening and sequence error checking. In effect the main function's purpose is to pass input time interval records from the input object to the output object.

The reader should examine carefully the use of static declarations in the **tlinput.c** and **tlmgout.c** source files. Such declarations are the means of controlling the "public" versus "private" issues for both data and functions associated with an encapsulated object in ANSI C.

```
/*** tlalgtyp.h ***/
  typedef enum{false,true}BOOLEAN;
  typedef struct{unsigned int start_time,end_time;} TIME_REC;

/*** tlinput.h ***/
  void initialize_input_files(void);
  BOOLEAN input_file1_rec(TIME_REC *);
  BOOLEAN input_file2_rec(TIME_REC *);

/*** tlmgout.h ***/
  void initialize_output_file(TIME_REC r1, TIME_REC r2);
  void output_merge(TIME_REC time_rec);
  void wrap_up(void);

/***Function prototypes recorded in util.h***/
FILE *my_open(const char prompt_message[],
              const char open_attributes[], char phys_file_name[],
              const int max_name_length);
BOOLEAN error_check_data(TIME_REC *previous_record,
                         TIME_REC current_record);


/*** mgmain.c ***/
/*** Main function for a time line algebra merge.  Two input files
    containing time interval records ordered on start time are
    co-sequentially processed to produce a single output file of
    merged records.  The essential difference from common
    "match/merge" logic of co-sequential file processing problems is
    that inter file time records (but not intra file time records)
    may overlap thus resulting in additional logical requirements
    over the simple output of the former.
      The input data files and the output data file are treated as
    encapsulated  objects, implemented in separate source files.
    Only "public" functions declared in those objects are permitted
    to manipulate the file objects.
      The Van Doren synchronization model for co-sequential
    processing forms the basis for the implementation.  ***/
#include "..\tlalgtyp.h"
/****    The above header file contains the following declarations:
  typedef enum{false,true}BOOLEAN;
  typedef struct{unsigned int start_time,end_time} TIME_REC;  ****/
#include "..\inputdir\tlinput.h"
/***    The above header file contains the following declarations:
  void initialize_input_files(void);
  BOOLEAN input_file1_rec(TIME_REC *);
  BOOLEAN input_file2_rec(TIME_REC *);  ***/
#include "tlmgout.h"
/***    The above header file contains the following declarations:
  void initialize_output_file(TIME_REC, TIME_REC);
  void output_merge(TIME_REC);
  void merge_flush(void);  ***/
```

```
void main(void){
  /**Basic data structures**/
  TIME_REC time_data1, time_data2;
  BOOLEAN more_input_exists;

  /*Shorten up reference to start time key fields*/
  unsigned int *start1, *start2;
  start1 = &(time_data1.start_time);
  start2 = &(time_data2.start_time);

  initialize_input_files();

  /***Prime the pump with initial input records***/
  more_input_exists = input_file1_rec(&time_data1);
  more_input_exists = input_file2_rec(&time_data2);

  /**Initial start times are required to initialize output
      buffer**/
  initialize_output_file(time_data1,time_data2);

  /*Principal synchronization loop which runs so long as input
      exists in either input file. */
  while(more_input_exists == true){
    if(*start1 < *start2){
      output_merge(time_data1);
      more_input_exists = input_file1_rec(&time_data1);
    } else /* *start2 <= *start1 */ {
      output_merge(time_data2);
      more_input_exists = input_file2_rec(&time_data2); } }
  /*Must flush the merge buffer as the last output record, if any*/
  wrap_up();
} /*end main*/


/****tlinput.c****/
/****  This source file encapsulates a pair of ordered input files
    intended for input to a cosequential time line algebra process
    such as time line merge or time line intersection.  The intent
    is to treat these files as an encapsulated object with strict
    control over "public" and "private" functions and data. ****/
#include <stdio.h>
#include <limits.h>
#include <stdlib.h>
#include "..\tlalgtyp.h"     /*** time record type definition***/
#include "..\utildir\util.h" /***Prototypes for utility functions
                              error_check_data and my_open ***/
/***The following is included here for compiler verification of the
    "public" function prototypes against the actual definition to
    defend against this source file and "tlinput.h" from getting
    out of synchronization.  ***/
#include "tlinput.h"
```

```
/*** Contents of tlinput.h
  void initialize_input_files(void);
  BOOLEAN input_file1_rec(TIME_REC *);
  BOOLEAN input_file2_rec(TIME_REC *);  ***/

#define MAX_NAME_LENGTH 30
/*The following is for max unsigned int "high value"*/
#define HIGH_VALUE  UINT_MAX

static  FILE *infile1, *infile2;
static  char phys_infile1[MAX_NAME_LENGTH],
           phys_infile2[MAX_NAME_LENGTH];
  /*Note initializers below*/
static  TIME_REC time_data1 = {0,0},  time_data2 = {0,0},
          prev_time_data1 = {0,0},  prev_time_data2 = {0,0};

void initialize_input_files(void){
  /*Open the two input files and the output file, binary mode*/
  infile1=my_open("Physical name of 1st binary input file-> ",
                "rb", phys_infile1,MAX_NAME_LENGTH);
  infile2=my_open("Physical name of 2nd binary input file-> ",
                "rb", phys_infile2, MAX_NAME_LENGTH);
} /*end initialize_input_files*/

/***This function is not visible outside this source file***/
static BOOLEAN input_time_rec(FILE *infile, TIME_REC *in_rec,
                    TIME_REC *prv_rec, TIME_REC other_file_input,
                    char file_name[]){
  static TIME_REC high_value_rec = {HIGH_VALUE , HIGH_VALUE};

  if (fread(in_rec,sizeof(TIME_REC),1,infile)>0){
    if(error_check_data(prv_rec, *in_rec) == true){
      puts("ABORTING on sequence error in file");
      puts(file_name);
      exit(1);    }
    return true; }
  /*EOF or error, treat as former for time being*/
  if(other_file_input.start_time == HIGH_VALUE){
    fclose (infile1); fclose(infile2);
    return false;} /*Double end of file at this point so no more*/
  *in_rec = high_value_rec;
  return true;
} /*end input_time_rec*/



BOOLEAN input_file1_rec(TIME_REC *record){
  BOOLEAN more_input_exists;
  more_input_exists = input_time_rec(infile1,&time_data1,
                      &prev_time_data1,time_data2, phys_infile1);
  *record = time_data1;          /**Return the input record**/
  return more_input_exists;      /**Return the EOF condition**/
} /*end input_file1_rec*/
```

```
BOOLEAN input_file2_rec(TIME_REC *record){
  BOOLEAN more_input_exists;
  more_input_exists = input_time_rec(infile2,&time_data2,
                    &prev_time_data2,time_data1, phys_infile2);
  *record = time_data2;          /**Return the input record**/
  return more_input_exists;      /**Return the EOF condition**/
} /*end input_file2_rec*/
/***end of tlinput.c ***/


/****tlmgout.c****/
/**** Output file object. Time line merging of time records of the
     type commonly occurring in time line algebra problems. Note
     that the logic has been constructed so that proper operation
     will occur even if one or both of the input files are empty.
     This is of some importance when performing time line algebra
     in a complex environment.****/
#include <stdlib.h>
#include <stdio.h>
#include <limits.h>
#include "..\tlalgtyp.h"        /**Record type declaration**/
#include "..\utildir\util.h"    /**Sequence checking and opening
                                functions**/
/**** The header file below is included so the compiler will verify
     the function prototypes against the actual definitions to
     defend against the possibilty of inconsistencies. ****/
#include "tlmgout.h"
/****  Contents of tlmgout.h:
  void initialize_output_file(TIME_REC r1, TIME_REC r2);
  void output_merge(TIME_REC time_rec);
  void wrap_up(void); ****/
/***Note information access issue. These static global items are
   available in this source file but are not visible elsewhere
***/
static unsigned long out_count=0;          /* Output record count */
static unsigned long received_count=0;    /* # of records received
                                          for merging*/
static TIME_REC out_rec;                  /*Candidate output record*/
static TIME_REC prev_out_rec = {0,0};   /*For sequence checking*/
static char physfile[50];
static FILE *outfile;

void initialize_output_file(TIME_REC rec1, TIME_REC rec2){
  outfile=my_open("Physical name of the merged output file->",
                "wb", physfile,50);
  /*Establish the record with the earliest start time as the first
    candidate output record.  (The first invocation of output_merge
    will be redundant, whichever input record is chosen.)    */
  if (rec1.start_time < rec2.start_time)  out_rec = rec1;
     else  out_rec = rec2;
} /*end initialize_output*/
```

```
void output_merge(TIME_REC time_data){
  /*Keep in mind that "time_data" is the earliest record, in terms
      of start_time, not yet processed.  Also keep in mind that
      "out_rec" MUST have been composed from earlier data! */
  received_count++;
  /***Defensive error check because of logical complexity***/
  if(time_data.start_time < out_rec.start_time){
    puts("Record merging ABORTING on merging data error to file");
    puts(physfile);
    exit(1);    }
  if( out_rec.end_time < time_data.start_time){
    /*No further merging can occur with data in "out_rec"*/
    /*Must output the candidate record and
          reinitialize with time data*/
    /*Defensively check it, because of complexity of logic*/
    if(error_check_data(&prev_out_rec,out_rec)==true){
      puts("Record merging ABORTING on sequence error");
      puts(" in output record to file");
      puts(physfile);
      exit(1);    }
    fwrite(&out_rec, sizeof(TIME_REC), 1, outfile);
    out_count++;
    out_rec = time_data;  /*Establish new candidate output record*/
    return;    }
  if( time_data.end_time <= out_rec.end_time){
    return;    }     /*Can't extend to the right so nothing to do*/
  /*Overlap exists on the right so extend "out_rec"*/
  out_rec.end_time = time_data.end_time;
} /*end output_merge*/

/**This function name is not visible outside this file**/
static void merge_flush(void){
  /*Flush merge buffer */
  /*Defensively check it, because of complexity of logic*/
  if(error_check_data(&prev_out_rec,out_rec)==true){
    puts("Record merging ABORTING on sequence error");
    puts(" in output record to file");
    puts(physfile);
    exit(1);    }
  fwrite(&out_rec, sizeof(TIME_REC), 1, outfile);
  out_count++;
} /*end merge_flush*/

void wrap_up(void){
  /**Guard against empty input yielding no output**/
  if(received_count > 0) merge_flush();
  fclose(outfile);
  printf("%lu records merged into ",  received_count);
  printf("%lu records written to file %s\n", out_count, physfile);
} /*end wrap_up*/
/*** end of tlmgout.c ***/
```


139

```
/*** util.c ***/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "..\tlalgtyp.h"
#include "util.h"
/***   Function prototypes recorded in util.h
  FILE *my_open(const char prompt_message[],
          const char open_attributes[], char phys_file_name[],
          const int max_name_length);
  BOOLEAN error_check_data(time_rec *previous_record,
                      time_rec current_record);    ***/

FILE *my_open(const char prompt[], const char attr[],
            char file_name[], const int name_size){
  FILE *local;
  fputs(prompt, stdout);
  fgets(file_name, name_size, stdin);
  /*Take care of extranenous newline*/
  file_name[strlen(file_name)-1] = '\0';
  if((local = fopen(file_name, attr))==NULL){
    printf("Can't open file named ->%s; aborting\n",file_name);
    exit(1);    }
  /*Caller needs the file structure pointer delivered by "fopen"*/
  return local;
} /*end my_open*/

BOOLEAN error_check_data(TIME_REC *p_data, TIME_REC c_data){
  BOOLEAN local_error = false;

  if( p_data->end_time > c_data.start_time){
    local_error = true;
    printf("End time ->%u of previous record",p_data->end_time);
    printf(" exceeds start time->%u of current record\n",
           c_data.start_time);    }
  if(c_data.end_time < c_data.start_time){
    local_error = true;
    printf("End time occurs before start time\n");   }
  *p_data = c_data;  /*Copy current record to previous for next
                       iteration*/
  return local_error;
} /*end error_check_data*/
/*** end of util.c ***/
```

The time line merge program presented above represents a non-trivial application of co-sequential processing logic. Sample assignments would be to give the students the code included here and require that they revise it to a time line intersection program, or vice versa, or give them the code in a non-objective form and require that it be revised to the form presented here.

SUMMARY

Properties of file I/O in ANSI standard C have been elaborated in this paper and illustrated with two programs, a hex dump program and a time line co-sequential merge program. Assignment variations of these types of programs can provide useful reinforcement of several important notions in teaching file structures, namely: binary files, co-sequential processing, and encapsulation of file objects.

REFERENCES

1. ANSI C, A Lexical Guide, by Mark Williams Company, Prentice-Hall, 1988

2. Applications Programming in ANSI C, by Richard Johnsonbaugh and Martin Kalih, Macmillan, 1990

3. C, A Reference Manual, 3rd edition, by Samuel P. Harbison and Guy L. Steele Jr., Prentice-Hall, 1991

4. C How To Program, by H.M. and P.J. Deitel, Prentice-Hall, 1992

5. C: The Complete Reference, 2nd edition, by Herbert Schildt, Osborne McGraw-Hill, 1990

6. C Traps and Pitfalls, by Andrew Koenig, Addison-Wesley, 1989

7. File Structures, 2nd edition, by Michael J. Folk and Bill Zoellick, Addison-Wesley, 1992

8. Programming in ANSI C, by Ram Kumar and Rakesh Agrawal, West, 1992

A C.A.S.E. for Systems Implementation

Paul Moriarty
North Country Community College
20 Winona Avenue
Saranac Lake, NY 12983-0089

I have taught a two-course sequence in Systems Analysis and Systems Implementation at Borough of Manhattan Community College (CUNY) for several semesters. The purpose of this paper is to outline a method of software development used in the second course of that sequence: Systems Implementation. The software used allowed an elementary form of CASE (Computer Aided Software Engineering) to be used rather effectively.

Systems Analysis

DAT 330, the Systems Analysis course was a three-credit, three-hour course without a laboratory. I mention this, because most of the courses in the Data Processing department consisted of lecture and hands-on laboratory sessions. This course was different in that respect, and both students and instructors enthusiasm seemed dampened in both taking and teaching this course.

The text used was Whitten, Bentley and Ho's System Analysis and Design, published by Richard D. Irwin. I feel it was an excellent book, although some other instructors did not agree with me. The course emphasized the System Life Cycle, and design procedures. Particular emphasis was placed on file design and organization along with input document design, output report design, and screen design.

The course was not entirely theory, however. Students were assigned the project of designing a business system on their own. I prepared several case studies, and students could pick one to work on. One of the case studies suggested was to design a system for a video rental application. Needless to say, this was a popular choice. Less popular was an inventory system, and another was a distribution system.

For the practical design exercise, students were to put together as a group to design a document containing file layouts for all the files in the system: data files, reference files, etc. In addition, for each of the files a screen layout for input and editing of data was to be done. Printer spacing charts were used to design and specify the reports to be generated by the system. The final document was rather large when done well. Students were cautioned not to let the project lag, as it would be impossible to complete the assignment in one or two sittings.

Students were encouraged to use a text editor or word processor to capture their work so it would be easily revised and shared among the group members. In order to ensure that there would be no procrastination and resultant disasters, milestones were established for the completion of stages of the project. Each milestone was handed in, corrected and returned to the group, heavily annotated as to errors or alternative ways of doing things. By the time the project was due for final revision, students had a good idea of the kind of material wanted.

A word here about the student groups: I allowed students to form groups of no more than three people. Two was ideal. The reason for this is that the project is rather lengthy, and may overpower one person. Two can share the work, and each can take on the chores they may have strengths in tackling. Three is less ideal a number. For some reason two people usually take on all the work, and the third person is ostracized, or left ineffective.

I am telling you about the first course, because the work done here is directly related in many cases to success of the second, even though both courses also stand alone.

I might mention here that adoption of the Whitten text allowed us to obtain a site license for Excelerator, a commercial CASE tool. Unfortunately, we did not have the hardware to support the software. Since then, a

laboratory of PS/2's with hard disks has been installed at the college, and this is now available to faculty and students.

Systems Implementaton

The second course in the sequence, Systems Implementation, has the goal of enabling students to program and implement their design from the first course, or a new design of their choosing.

The problem with this course was how the students could get a viable system put together with the tools at hand. In the past, COBOL, Pascal and Basic were the only languages available. Development time for a system written with these languages would take too long, certainly more than a semester.

To overcome the language problem, and to expose the students to a product not covered in their other classes, I chose to adopt dBase III+ for the implementation course. For the amount of code written, dBase would certainly get more done in less time than would COBOL or Pascal. dBase has other advantages: file creation, screen design, report development are all done quickly and much more easily.

One day, while talking to my McGraw-Hill representative, she made a passing reference to Genifer, a dBase development tool. I was familiar with Genifer as a commercial code generator. When she told me that there was a way I could get the product for free, my ears really perked up. The deal specified that if we adopted Systems Development Projects, by William and Camille Amadio, we could get the latest release of Genifer from Bytel gratis.

We adopted the text, filled out a site license for Genifer and sent it to Bytel. Genifer arrived shortly thereafter. Expecting a watered-down student edition, I was surprised to find that we had received the full commercial version. McGraw-Hill and Bytel are to be applauded for making this possible.

With the text, there is also available an abbreviated Genifer 2.0 user's manual. Students purchase both books, which are shrink-wrapped together. McGraw-Hill provides an educational version of dBase III+, so all of the software you need is given, and free.

I must point out here that if one wanted to get technical, Genifer is not a CASE tool, but a code generator. That point can be made, and I'll agree with you and add that it is really a template language code generator. Students must, however, enter all of the specifications for their project into Genifer to reap the benefits of having the code written for them. They do not do the design work in Genifer, but the implementation is greatly aided by using Genifer. Isn't that what CASE is all about, getting the code written? Students are also amazed that one can get a computer program to write computer programs.

WHAT GENIFER PROVIDES:

Genifer allows the students to create a data dictionary for database field names. It can also import a file created through dBase. In addition to the field names and data types, there are provisions for validation by range, list, file, and expressions. Index files for each data file are also specified with their key fields.

Screen design capability is supported for menus, data screens, and help screens. The menu screens allow selection of further sub-menus, help files, procedure files (dBase .PRG programs), or DOS commands.

Reports can be specified with headers and footers at the report, page, and control break levels, in addition to detail lines. Expressions for report fields can be used.

The program generation done by Genifer will produce menu, file maintenance, inquiry and report programs. Genifer templates are modifiable. If you do not like the code generated, changing the template will change all programs generated through that template. You can also add your own set of templates. Dialects of "XBASE"

are also supported through additional template sets available from Bytel. There are template sets for FoxPro, Clipper and dBase IV.

HOW IT WAS USED:

Students used Genifer to define all data and index files for their system. They designed the data screen for each file's maintenance program, in which data entry, additions, changes and deletions were done. The maintenance programs themselves were generated with Genifer.

Report programs were written by the students. Most of these were simple file listing programs. A model program from Alan Simpson's dBase III Plus Programmer's Reference Guide was provided and explained to the students.

For the video system, one update program of some complexity was hand crafted. This program was a transaction processor which updated three files. The transactions processed were video rentals and returns. The files requiring update were a video master, customer file, and sales file.

Once individual programs were completed, the students incorporated all of the modules together to produce a menu-driven system. The menus were generated by Genifer.

Once the system was ready, each student group had to demonstrate their system to the rest of the class. I provided an LCD/projector/computer setup that allowed real-time presentations. We also had available a huge color monitor which could be viewed by everyone in the room.

Conclusion:

Students received a great deal of experience in this course. They had to cooperate with each other, sharing the responsibilities and work required to get a functioning system together. They also had to sharpen their presentation skills and overcome (in some instances) severe stage fright. They also worked under deadline pressures and stress. Their clamoring for laboratory time progressed from mild requests to demands that the laboratory be made available for additional hours. Time management and work scheduling were no longer buzzwords.

They learned how to design balanced and legible screens, which was not covered in any other course offered. They compared different groups' designs, and adopted what they liked, and discarded what didn't appear well. They made errors, generated error messages, and ran and ran and ran their programs to eliminate bugs.

Students also learned how to do some coding in dBase, and realized some of the benefits of using a fourth generation language. In addition, they learned how to design a system and implement it. They also learned that computer programs can not only write other programs, but do it a lot better than programmers can for mundane programs.

Despite many problems and frustrations that they encountered, the students did well. Many systems ran well, yet there were some that were not successful. Everyone did experience some satisfaction at having worked at a difficult project and were happy to present their work. They were also amazed at the amount of work that they were able to accomplish in one semester.

As an aside, I might add that at Pace University's Master of Computer Science program there is a two-semester course sequence in Software Engineering, which was similar to this two semester course sequence. Systems written in that course were more ambitious, but the success rate was as good as mine.

HARDWARE REQUIREMENTS

If you read the hardware requirements in the Genifer manual, they will tell you that you need so much hard disk space. Don't believe it. I was able to get Genifer running on PS/2's with two 720KB floppy disk drives. All of Genifer's software was on the A: drive, and the student's data disk was in the B: drive. Students did have to do all of their work in Genifer, and then quit Genifer and use the dBase disk in the A: drive. There was some swapping of diskettes involved, but it did work.

You do need at least 512 KB of RAM. I doubt very much if you could get Genifer to run on 360 KB floppies. Of course, if you have hard disks or a network installation, these requirements are certainly not a problem.

One tool that I have found to be indispensible for teaching this course was a computer driving an LCD panel on an overhead projector. This equipment allowed me to demonstrate how to use Genifer and dBase very effectively. Students could see the result of keystrokes, and see the screen displays that they would be using.

COURSE DESCRIPTIONS

DAT 330 -- Systems Analysis 3 hours, 3 credits

This course builds on the knowledge students have gained in their programming courses. It teaches students to analyze the interaction between "state of the art" computer hardware and software and the tools and methods used by systems analysts to develop systems for computer applications. Students follow up on the theory by preparing a runbook which is an analysis and design of complex problem-stressing data format, documentation and operating instructions.

Prerequisite: DAT 205 (COBOL) or Departmental approval.

DAT 400 -- Systems Implementation 5 hours, 3 credits

Students continue from the realm of theory taught in DAT 330, Systems Analysis, to the realities of practical applications. The class is divided into teams. A system is developed as a joint effort by each team as it analyzes, systematizes, programs and writes documentation to implement its projects. In addition to the team projects, topics relevant to current data processing techniques are discussed and where applicable, demonstrated to or practiced by the class.

Prerequisite: DAT 305 (COBOL II) and DAT 330 or Departmental approval.

The Evolution of the Information Systems Organization

Thomas A. Pollack
Duquesne University
A.J. Palumbo School of Business Administration
600 Forbes Avenue
Pittsburgh, PA 15282-0104

Introduction

Information systems organizations have undergone a significant transformation in the last twenty years. Solutions, customarily developed by a centralized information systems department, now frequently originate as third party packages with end-user modifications. In many organizations, end-user computing has provided needed relief from request backlogs. As efforts are focused on moving from machine-centered computing to a user-centered approach, reliance and emphasis is placed on tools such as distributed networks and personal productivity software. An accompanying process for dealing with systems integration and end-user training is also essential.

As educators or information system administrators preparing for computing in the 1990's, we must ensure that training programs for end-users includes a "toolkit" comprised of at least seven critical components. In this paper, these components and their significance will be addressed.

Components of the "Toolkit"

In a lecture to MIS students at Duquesne University, Les Shindelman, Partner in Charge of MIS Consulting at Coopers and Lybrand, identified the following components of the end-user toolkit for the 1990's:

- Relational database
- Spreadsheet
- Word processor
- Fourth generation language
- Graphics
- Communications
- Micro/mainframe link

Although we may differ on the contents of the toolkit, conceptually, we probably believe in the development of a "common body of knowledge" for the professional of the 1990's. Some items of the toolkit may be discipline-specific. As competent educators, we are in a position to directly or indirectly develop these competencies through coursework. It is the "end product" knowledge level that ultimately concerns us.

Several components of the toolkit are obvious for all concerned. Competency in using word processing, spreadsheet and database software is generally taken for granted. Of course, we can discuss particular packages, but basic transfer of learning allows individuals to learn the conceptual framework of each of these software types. Many are convinced that the best route is an integrated package. Many of today's computer-related publications devote considerable effort to critical assessment of software packages (Carleton, 1991; Crawford, 1991; McCartney, 1991; Robins, 1992, et al.).

Many of the points made in the reviews are a significant help in software selection. Ease of use and power are always key concerns and significant advances have been made with the current emphasis on graphical user interfaces (GUI) such as Windows.

Although not as paramount as the software already mentioned, a quality graphics software package is also essential. Graphical representation of data has become an expected norm. Closely related and often

included in this category are versions of presentation software. Some even wish to include desktop publishing packages in this category.

Significant productivity increases can be realized through the use of fourth generation languages (4GLs). The query languages associated with 4GLs can open new avenues of information as they allow the user to gain direct access to existing databases. The report generation component provides a fast, simplified technique for producing output. One or more 4GLs are useful toolkit components.

The final area for discussion involves various types of communication packages. Certainly, the use of a local area network (LAN) has become somewhat commonplace. Electronic transfer of data, reports, messages, and mail has become a normal mode of operation. Use of external services such as Internet is also understood to be a common communications medium and is expected. It is important that end-users become familiar with one or more communications packages.

The micro to mainframe link is critical for data sharing. It is essential that end-users have access to data and also have a means to transfer data. Uploading and downloading files are critical operations in the Information Age. It is quite common to do processing on a microcomputer with data downloaded from a host computer. The host frequently acts as a database server and the microcomputer as a client sending queries. A convenient methodology to carry out these tasks is essential.

There are other categories of software that we might consider as critical components of the toolkit. We must be flexible in our thinking as we consider the contents of the toolkit. The Information Age environment seems to change almost daily. Changes in the content of our toolkit should in some way carry over to our curriculum if we want our students to be as well-prepared as they can be. This is frequently a huge challenge, but one we must be willing to tackle. We must guard against being caught in a situation with our education programs where we reward A and hope for B! The curriculum must be dynamic and responsive.

End-User Computing

End-user computing is a burgeoning phenomenon which allows users to develop, use and control their own information systems. According to Zwass (1992), although there are varying degrees of sophistication, three dimensions will be considered.

1. A sophisticated level of computer technology use (operation of a system).
2. End-user **development** of application systems, often with assistance from MIS professionals.
3. **Control** of information resources and the decision-making authority to acquire and operate information systems.

Leading to the emergence and growth of end-user computing are several key factors. Included are technological factors such as the power of the personal computer and the abundance of easy to use productivity software. Organizational factors such as improved responsiveness and effectiveness resulting in a superior competitive posture are important. Finally, the ability of personnel to participate in a process and realize productivity also serves as a positive influence (Zwass).

Included among the disadvantages of end-user computing are the following:

1. Ineffective Use of Financial Resources
2. Inappropriate Selection of Hardware and Software
3. Threats to Data Integrity, Security, Privacy
4. Errors in Systems Analysis
5. Lack of Quality Assurance in Systems Development and Operation
6. Proliferation of Private Systems (Zwass)

Although end-user developed systems cannot replace MIS professionally-developed systems for many types of applications, the systems have filled a niche in today's organizational environment. Despite shortcomings and problems, end-user systems will continue to grow in importance.

Implications for Higher Education

Many in the higher education community are accustomed to building "silos" and referencing these silos by such terms as computer science or information systems. The trend in the corporate community has shifted from large centralized mainframe based information system centers to decentralized, end-user driven service centers. In higher education a shift in emphasis has also occurred. Whereas ten years ago, only students specializing in a computer discipline developed a high degree of computer literacy, today a relatively high degree of computer literacy is expected of all students. Not only must educators be concerned about the computer competency of students with computer majors, but concern for the computer literacy of the overall student population must be a paramount concern. Everyone has become an end user. In a recent address to students and faculty of Duquesne University, John Staley, President of Federated Investors, indicated that the posture taken in recruiting entry level employees is that the prospective employee will have an equal or higher degree of literacy than the current workforce. The question, then, is how can institutions of higher education ensure that their students are adequately prepared? Programs and competencies will differ, but the general objectives will be the same.

Allow me to describe the plan I and the MIS faculty have devised for both students and faculty at Duquesne University's School of Business Administration. Students entering the Business School are required to take two computer-related courses in their freshman year. The first course is very application-oriented and is entitled 182 Microcomputer Software Applications. The content includes projects which require use of word processing, spreadsheet, and graphics software. The second course, entitled 183 Management Information Systems is more theoretical in nature. In addition to explaining the use and value of information systems in organizations, the course includes projects which reinforce the use of previously learned software while also requiring use of database software. Our objective in requiring the two courses in the freshman year is to provide a computer literacy foundation and facilitate the integration of meaningful computer applications across the business disciplines.

As agreement is reached on the content of the "toolkit", each of these software applications will be embedded within the framework of the two required courses. This will prepare students to be knowledgeable end-users. In addition to the actual coursework, MIS faculty will conduct workshops which present each of the toolkit topics to the faculty and staff of the School. This will serve to broaden the computer literacy base of faculty, and should contribute to more meaningful and relevant integration by discipline.

Many institutions have not taken the necessary steps to move their faculty and staff to the next level of performance with regard to computers and information systems literacy. The three stages of development (Reynolds, 1988) in the use of technology is worth considering as we analyze this problem. The introductory stage involves use of stand-alone, single
purpose applications such as word processing. The integration stage usually involves networking and machine to machine communication which focuses on the improvement of productivity for professionals and managers. In the advanced stage of development, the objective is to improve the overall productivity and job satisfaction of all involved and graduate from stand-alone operations to entire processes.

Many faculty, and as a result, students have not graduated to the next higher level of performance in their computer usage. Many need to be guided in this development process. As with system implementation, a well-organized and comprehensive training program is most important. By moving faculty to the next level of competency, it is reasonable to assume that, in most cases, students will be beneficiaries. A word of caution about a trap to which many of us fall victim: Do not assume that the next level will come naturally for all concerned!

Conclusion

Several conclusions can be drawn from what is stated in this paper. First among the conclusions is the fact that end-user computing will continue to grow and assume a more prominent role in organizations in the 1990's. Secondly, end-users need a repertoire of tools in the form of a computer competency toolkit. The actual tools to be included may vary somewhat by organization. A degree of consensus on the key components already exists.

Thirdly, educational institutions and particularly computer-related faculty must continue to assume roles of leadership to assure that the level of overall literacy continues to increase. As educators, we must continue to examine, refine and share what we do. Finally, the end result of adding structure and quality training for computer competencies will be a more competent end-user; one better prepared to cope with the evolving changes in computers and information systems.

References

Carleton, Mary & Russ (1991, November). Network utilities: building the LAN toolkit. Infoworld, pp. 59-78.

Crawford, Walter (1991, December). Desktop publishing choices: making an appropriate decision. Information Technology and Libraries, 311-316.

Kaplan, James M. & Evart, Bob (1991, April). Saying things right! Internal Auditor, pp. 16-19.

Kroenke, David (1992). Management information systems. New York: McGraw Hill.

Lombardi, John (1991, October). Grammar checkers: keep your writing in line. Infoworld, pp. 64-77.

McCartney, Laton (Ed.) (1991). Bridging the innovation gap: nine technologies for the 1990s. InformationWeek.

Reynolds, George W. (1988). Information systems. USA: West.

Robins, Gary (1992, January). Off-the-shelf PC: retailers list software choices. Stores, pp. 105-110.

Shindelman, Les. Lecture, Distinguished Speaker Series, Duquesne University, October, 1990

Staley, John. Lecture, Business Leader Forum, Duquesne University, April 22, 1992.

The writer's toolkit: more than just your average grammar checker. (1991, October). Infoworld, p. 80.

Yee, Dianna Ann. Famous last words. Unix Review, 121-124.

Zwass, Vladimir (1992). Management information systems. USA: Wm C. Brown.

Out from the Swamp: The Evolution of a Software Engineering Course

Robert L. Sedlmeyer
Department of Computer Science
Indiana University -- Purdue University at Fort Wayne
BITNET: SEDLMEYE@IPFWCVAX

ABSTRACT.

Through the generosity of several corporate sponsors a 20 station PS/2 lab was created in 1988 for the School of Engineering and Technology. The Department of Computer Science utilizes the lab for a number of service courses as well as its Software Engineering course. We describe how the lab is configured, managed and utilized to support the Software Engineering course. Specifically we discuss our (still evolving) approach to integrating Computer-Aided Software Engineering (CASE) into the lecture and project components of the course through demonstrations, guided laboratory exercises, and traditional out-of-class assignments.

INTRODUCTION

The goal of computer science education at IPFW is to impart to the graduate the knowledge and skills necessary to analyze, design, implement and evaluate computer solutions to real-world problems. Our Department offers two- and four-year degree programs in the areas of Computer Science and Information Systems. The Computer Science degree is designed to prepare students as computer science professionals as well as for possible graduate study. The Information Systems degree is designed to prepare students as software developers and managers of business systems applications. Both degree programs have a common core of required courses that are based on the ACM Curriculum 78 guidelines [1], and both emphasize the practice of software engineering principles.

Our curriculum strives to balance theory and practice: We aim to create graduates who display competence in both discipline-oriented thinking and tool use, two competency areas identified in the "Report on Computing as a Discipline." [2] Concerning tool use the report states that "majors must be familiar enough with the tools to work effectively with people in other disciplines to help design modes of effective interaction in those disciplines." Since the emergence of CASE technology in the mid-1980's many employers in our service area have incorporated CASE tools into their software development processes. As a consequence, employer expectations of CASE experience have risen significantly, and we feel an increasing urgency to incorporate such tools more broadly and deeply within our curriculum in order to prepare our graduates to enter the workplace.

Beyond meeting the needs of potential employers we see three pedagogical benefits of CASE. First, CASE can guide the student in following a methodology or enforce the discipline of that methodology. Secondly, CASE allows the student to focus on the more creative aspects of software development while letting a tool handle the more clerical and repetitive tasks. Finally, CASE can support the development of artifacts which can be used to teach software engineering topics or serve as the basis for laboratory assignments across the entire life-cycle. CASE may also provide a practical benefit: Granger and Pick [3] found that students who used CASE produced more complete systems in less time than those who did not use CASE.

To maximize the benefits of CASE we believe that it should be integrated within the entire curriculum, but this is not an easy task. CASE technology is expensive and most CASE tools have steep learning curves. We have developed a comprehensive plan for inserting CASE technology into our curriculum and have submitted a proposal to NSF for funding its implementation. Our plan consists of four steps. First we will analyze our current curriculum to determine placement and coverage of software engineering topics. We will then identify and evaluate CASE tools to support activities associated with each topic. This will culminate in the specification of a software engineering laboratory (The funds for the laboratory will be sought in a companion proposal.) Finally,

we will design and test a set of curriculum modules for teaching how to use particular CASE tools to support particular activities.

In formulating our CASE-insertion strategy, we have drawn upon our experiences in teaching software engineering. Since 1987 we have been using CASE in CS 360, a junior-level software engineering course. In this paper we describe the software engineering course and the laboratory which supports it, discuss how CASE has been integrated into the course, and share some of the lessons we have learned over the last five years.

COURSE ORGANIZATION

The objectives of CS 360 are to introduce the student to the software engineering process and to gain experience with tools which support that process. We utilize the classic waterfall model to organize course topics. Lectures covering requirements analysis, functional specification and design emphasize the Modern Structured Analysis/Structured Design methodology [4, 5]. The textbook is Software Engineering with Student Project Guidance [6] by Mynatt. The text is supplemented by a reading list. To support the software engineering process we introduce the students to three tools: Excelerator, which supports analysis and design; the VAX Screen Management Run-time Library, which supports the implementation of a window-based interface; and the VAX Performance and Coverage Analyzer, which supports the testing phase.

Lectures on software engineering topics and tools are put into practice in a semester-long team project. The team project has several objectives. The first is to give students insight into the need for an engineering approach to software development. For many students the project is the first program which forces them to think carefully about requirements and design before plunging into coding. The complexity of the project also instills an appreciation for tools. Thirdly, students are baptized into team programming. This experience produces a keen awareness of the non-technical factors that affect programming-in-the-large, and introduces students to the dynamics of group problem-solving. Finally, the project demonstrates that successful software development requires communication skills as well as programming skills. Students must also use oral and written communications skills to create project documentation and in-class presentations.

PROJECT ORGANIZATION

On the first day of the semester we describe the application area and hand out a document containing the preliminary project requirements. We have found that instructor-defined projects best facilitate the meeting of course objectives, but we have strived to make the projects realistic. Projects have included a bibliographic retrieval system, a microcode simulator, a forms designer, and a student tracking system for computer science majors.

During the first week we VAX MAIL a survey to the class to obtain background information for team formation. We then create teams containing from three to five students attempting to equally distribute knowledge, skills, and experience. When possible, student preferences for teammates are honored. This reduces the energy required for team maintenance and helps maintain morale when the going gets tough. We allow the students to create their own team organizations but require one person to be designated as the official contact person. This person serves as an interface between the instructor and the team for communication of technical and non- technical issues, scheduling of meetings, and distribution and collection of project deliverables.

The project itself is divided into a sequence of milestones. Each milestone is accompanied by a set of deliverables. Deliverables consist of both documentation and in-class presentations. Table 1 summarizes the project milestones and deliverables.

Milestone	Deliverable
Preliminary Specification	Environmental Model
Final Specification	Behavioral Model Walkthrough
Interface Design	Interface Design Document Walkthrough
Software Architecture Design	Software Arch Design Document Design Review Preliminary User's Manual
Implementation	Source Code Test Documentation System Demonstration User's Manual

The environmental and behavioral models are represented as data flow diagrams and data dictionaries. Also included are the refined project requirements. The interface design document consists of a syntactic model (menu, form, and report layouts) and semantic model (user-oriented task sequence for each system function). Hierarchic structure charts and pseudocode define the software architecture. Test documentation follows the IEEE standard [5]. Formal testing is required for only a small subset of the system. Both white-box and black-box testing methods are utilized.

Each deliverable is evaluated for correctness, completeness, consistency, and clarity. While most teams are successful in completing each milestone, each semester has its painful moments. We address the reality of performance disparities among teams and individuals within a team in two ways.

To promote the success of each team we develop baselines for the first four milestones. Instructor effort is considerable, but may be diminished by modifying a team's deliverable. Baselines have a number of benefits. Foremost, they allow a team to recover from disaster, especially if it occurs early in the project. Secondly, we can assign the next milestone upon receipt of deliverables for the current milestone rather than waiting until these deliverables have been evaluated. Finally, baselines give students software engineering artifacts by which they can assess previous efforts and which can guide future efforts.

To fairly judge individual effort for partially completed milestones each team must file two reports: a workload distribution plan and an effort report. The distribution plan outlines team expectations for each member. We offer significant guidance here to ensure an equitable distribution of effort. The distribution plan must be completed one week after milestone requirements are handed out. The effort report details each member's actual contributions. Comparing these two reports we can identify those who were most responsible for the unfulfilled milestone requirements and reward those who performed heroically.

CASE HISTORY

In this section we chronicle the introduction of CASE into our software engineering course and discuss its impact on course content and student performance.

Prior to 1987 we did not have access to any CASE tools for support of analysis and design, but we did have several tools which supported the implementation and testing phases. As a result, the lectures and assignments

were unbalanced in favor of implementation. Three tools were covered: The VAX symbolic debugger, the VAX Librarian, and the VAX Screen Management Library. One week was spent on each tool and an additional week was taken up by related, but unscheduled, topics e.g. the VAX command language. Each tool was presented using a mixture of formal lectures and in- class demonstrations. For each tool a small assignment, unrelated to the project, was completed by each student in preparation for use within the project. The students enjoyed learning about these tools and profited especially from their introduction to the debugger. They were able to eliminate reams of paper produced by debug statements and reduce the total number of compilations required to produce functional code. As a result their productivity increased.

In 1987 we received an educational grant from Index Technology (now Intersolv) and obtained two copies of their CASE product, Excelerator. During 1987 we immersed ourselves in Excelerator and used it to document a software system for supporting graph theory research. This experience gave us confidence that Excelerator could be used profitably in CS 360, but its introduction was delayed due to lack of hardware resources. We could not justify requiring 25 students to compete for two Excelerator workstations.

By the fall of 1988 sufficient hardware resources were obtained. A microcomputer laboratory proposal, designed to support Engineering and Technology programs, was funded by donations from several local corporations. The laboratory contains 20 IBM PS/2 Model 50 microcomputers configured with 1M of memory and a 30M hard disk. Each PS/2 has an EGA color monitor and mouse. Software includes DOS, Harvard Presentation Graphics, Microsoft Word, Quick C, QuickBASIC, a number of discipline-specific applications such as SPICE, and, of course, Excelerator. Every four machines share an IBM Proprinter. A designated instructor's station is wired to a SONY projection system for in-class demonstrations. Recently, a LAN and file server were installed, and by next year the lab will be connected to the campus backbone network.

Since the spring of 1989 we have slowly incorporated Excelerator into CS 360. In the beginning we just used it to create the software architecture design, but since then have we expanded the scope of Excelerator to include analysis and interface design. This spring each of the first four milestones had deliverables produced and analyzed by Excelerator. To integrate Excelerator so fully has required careful managing of the laboratory, adjustments to course content, and project simplifications.

The laboratory is open to all students enrolled in Engineering and Technology courses. This has two practical consequences. First, we must share the hard disk space with all other departments in the school. Excelerator itself consumes about one-third of available disk storage, and each project requires an additional 3M. Fairness dictated creation of a single Excelerator project shared among all teams rather than a separate project for each team. Students are not permitted to store data or program files on the hard disk, and the lab manager enforces this rule with vigor. These two considerations forced each student to save the contents of the project dictionary (via the BACKUP utility) after each Excelerator session. On several occasions students forgot to perform backups and lost significant amounts of work. We now use a batch file to invoke Excelerator which automatically performs this backup.

A more serious consequence of the open laboratory is access. The laboratory is open from 9:00 AM to 9:00 PM weekdays and from 10:00 AM to 5:00 PM on weekends. Since most of our students work full- time only the evening weekday hours are available to them. Unfortunately, it is precisely these hours that are most highly sought by other instructors for in-class demonstrations or closed laboratory sessions. This past semester, as a result of several popular Continuing Education courses, almost half of the available lab hours were scheduled for these activities. After much wringing of hands the Computer Science department agreed to fund additional evening and weekend hours, but several students still resorted to using vacation days to complete their Excelerator work. We hope to resolve the access problem by creating a software engineering laboratory through an NSF Instrumentation and Laboratory Improvement grant.

We have discovered, through much tribulation, that the laboratory hardware and Excelerator software obey Murphy's Law. We are fortunate to have a lab technician that can deal with burned-out memory chips, misaligned disk drives, and jam-prone printers. It is of little consolation, however, to students whose latest backup cannot be read because of a faulty drive or whose session was terminated and project dictionary compromised

by a hardware failure. We can only sympathize and advise them to backup early, often, and on machines that have good track records.

One of the biggest headaches in managing the Excelerator software has been to maintain system integrity. System and project dictionaries can be compromised in a variety of ways, the most common being failing to exit the system under Excelerator control (e.g. just turning off the machine). While Excelerator provides utility programs to recover from most faults we have found it prudent to create additional safeguards. We protect Excelerator executable files by designating them as read and execute only. We maintain a backup directory with all files required to recreate the common Excelerator project. These files are used to initialize the system whenever Excelerator is invoked through our batch file. We train two or three students as system managers so that can correct minor problems without having to contact us. We are also developing a situation handbook which contains the most common problems and how to fix them.

The in-depth usage of Excelerator has required several adjustments to course content. Excelerator is a sophisticated CASE tool and has a steep learning curve. Even the subset of facilities that we utilize requires a significant investment of time to master. We decided that significant instruction time should be devoted to Excelerator at the expense of traditional course topics. We elected to remove discussion of the detailed design and implementation phases (except for integration). We reasoned that both of these phases already received considerable attention throughout our curriculum. This gained us three weeks of class time (6 meetings) which we distributed as follows: (1) overview of Excelerator functionality and interface; (2) drawing dataflow diagrams; (3) describing DFD entities; (4) report generation; (5) drawing structure charts and describing associated entities; and (6) screen design and execution. These topics were interweaved with software engineering topics. For example, when we discussed interface design we covered Excelerator screen design and execution.

We present each facet of Excelerator in the laboratory with a combination of lecture, demonstration and hands-on exercise. Each presentation lasts 75 minutes and is equally divided among the three pedagogic activities. The lecture serves to introduce the Excelerator function. Major features are highlighted, steps outlined, and connections to other functions are pointed out. In addition we describe what lifecycle activites are supported, although this is often obvious. Once the student has been familiarized with the purpose and context of the function a simple demonstration is performed. The demonstration emphasizes features relevant to project deliverables and is accompanied by a running commentary. Students are provided with a handout of the demonstration artifact. Often this handout shows the artifact in various phases of construction and contains both screen images and Excelerator output reports or graphs. A short question and answer period follows the demonstration and then the students are given a hands-on exercise. The exercise reinforces the lecture and utilizes features just demonstrated. Students are not required to complete the exercise in-class and it is not graded.

We recognize that one class period is not enough to make students proficient with any Excelerator function. Additional lecture time is made available at the beginning of each class period for addressing Excelerator issues. A complete set of Excelerator manuals are also provided in the laboratory along with a practical tutorial [7].

In addition to Excelerator we also cover two other tools: the VAX Screen Management Library and the VAX Performance and Coverage Analyzer. These tools are presented in the same way as Excelerator with the exception that the hands-on exercises are replaced by homework.

Use of Excelerator has also had an impact on project expectations. The emphasis of the project is no longer on the product but on the process. In the past at least half the semester was devoted to the coding phase; now analysis, design and implementation are each allocated one third of the semester (4 weeks). This emphasis is also reflected in the determination of the final grade: The source code deliverable contributes only 1/6 of the final grade.

As much as we preach about the importance of the process students are loathe to end the semester with an unfinished product. To balance course objectives with the students' need for accomplishment we have taken a

spiral approach to the project. During the analysis phase students must create a functional specification that addresses all user requirements. In each successive phase we reduce these requirements so that a finished product can be delivered without unnecessary burden. At the end of the semester each team can produce a working system which addresses a large, but not complete, subset of the original requirements. We have found this approach to be very well received by students.

SUMMARY

Software Engineering at IPFW has evolved from a product-oriented course to a process-oriented course with increased utilization of CASE tools. The introduction of CASE tools was made possible by receipt of an educational grant from Intersolv supplemented by modest Departmental funds and by the creation of a microcomputer laboratory. Integration of CASE has been a slow but steady process and has required modifications of course content and project expectations. With the insertion of CASE into the software engineering course we have begun modernizing our curriculum to address the needs of 21st century graduates.

REFERENCES

1. ACM Curriculum Committee on Computer Science, "Curriculum 78: Recommendations for the Undergraduate Program in Computer Science," Communications of the ACM, March, 1979, pp. 147-166.

2. P. J. Denning, D. E. Comer, D. Gries, M. C. Mulder, A. B. Tucker, A. J. Turner, and P. R. Young, "Computing as a discipline," Communications of the ACM, Vol. 32, No. 1, January, 1989, pp. 9-23.

3. M. J. Granger and R. A. Pick, "The Impact of Computer-Aided Software Engineering on Student Performance," SIGCSE Bulletin, Vol. 23, No. 1, March 1991, pp. 62-72.

4. E. Yourdon, Modern Structured Analysis, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

5. E. Yourdon and L. Constantine, Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design, Prentice Hall, Englewood Cliffs, New Jersey, 1979.

6. B. T. Mynatt, Software Engineering with Student Project Guidance, Prentice Hall, Englewood Cliffs, New Jersey, 1990.

7. H. D. Crockett, G. R. Hall, M. R. Wheeler, Excelerating Systems Analysis and Design, Boyd and Fraser, Boston, MA, 1992.

A Closed Laboratory in CS-1 Using Tutorials

Peter D. Smith
Mathematics Department
Saint Mary's College
Notre Dame, IN 46556
psmith@saintmarys.edu

Abstract

The recent changes in the CS-1 curriculum have made it difficult and, in fact, undesirable to cover programming language coding and debugging techniques in lecture. At Saint Mary's, a small, Catholic, liberal arts college for women in northern Indiana, we have written some interactive tutorials to help students learn this material on their own, either before or at the beginning of the closed laboratory session. This paper will describe the changes in the CS-1 lab environment which brought us to this point, discuss this tutorial experiment and its effectiveness, and also describe our closed lab assignments.

History

There was very little discussion of closed labs in introductory Computer Science courses before the work of Norman Gibbs and Allen Tucker with their 1986 proposal for a model curriculum for a liberal arts degree in Computer Science [1]. The only reference to laboratories in ACM Curriculum '78 [2] is an admonition that CS-1 should include laboratory sessions where a student can develop good programming techniques under close supervision. There was no suggested outline for these laboratory sessions, nor was there any indication of how they were to interface with the lecture material. At Saint Mary's and, I suspect, at other colleges the laboratory section of CS-1 prior to 1986 was used to complete programming homework assignments under little or no supervision.

While not mentioning laboratories specifically, the Gibbs and Tucker paper [1] lay the foundation for Computing Curricula 1991 [3] by emphasizing the importance of treating computer science as a science and introducing the student to the discipline of the science in CS-1. It was the 1988 Denning report on the core of computer science [4] that alerted many of us to the need for curriculum revision and the importance of laboratories for teaching and learning computer science. In section 8 of this report, the task force delineated the three purposes for laboratories: to provide concrete experiences to help students understand abstract concepts, to teach good design, programming, and documentation skills, and to provide an introduction to experimental methods. The report indicates the importance of carefully written lab assignments and of the presence of lab instructors to watch that each student follows correct methodology. It talks about the coordination of lab problems with lecture material and the importance of constructing lab assignments which can be completed during the lab session.

It was the Denning report [4] that introduced the 9x3 knowledge unit matrix and pushed for a breadth-first curriculum. This report, given at the 1988 SIGSCE meeting, triggered one of the most lively debates I have witnessed at a national meeting. There were many skeptics including several textbook authors who thought such an ambitious curriculum was impossible to implement. Others worried that greater emphasis on theory would increase the rate of attrition, already a problem for many computer science programs.

The next few years were to see further stormy SIGSCE curriculum sessions as the ACM/IEEE Joint Curriculum Task Force fleshed out the ideas introduced in the Denning report. The final result was the publication of Computing Curricula 1991 [3]. In this document each knowledge unit includes a section on Selected Laboratories which work to tie the lecture and laboratory experiences closer together. Section 6.3 in this report describes the role of laboratories and distinguishes between open and closed lab assignments. During the last two years the number of SIGSCE and ASCUE conference panels, tutorials, and workshops on designing

closed laboratories for CS 1 and 2 has mushroomed, a clear sign that computer science faculty are attempting to implement the new curricular guidelines.

Lab Organization Nomenclature

The first distinction between open and closed laboratories was made by Peterson and Miller [5] in 1985. They called a lab closed if students were required to complete the assignment during the scheduled and supervised lab period and open if they could leave the scheduled and supervised lab and complete the assignment on their own. They did not differentiate between the types of projects which could be assigned in each type of lab but instead concentrated on the logistics of laboratory organization and the beneficial effect on student grades from the closed lab experiment.

The Denning report [4] distinguished between the types of assignments suitable for the two types of labs. Although not mentioning open and closed laboratories by name, this report specified that the supervised laboratory assignments should be coordinated with the lecture material and serve the three purposes mentioned in the History section above. The organized laboratory sessions were to supplement, not replace, the usual programming assignments.

The informal, semi-formal, and formal terminology for computer science laboratories was introduced by Parker, Cupper, Keleman, Molnar, and Scragg [6] in 1990. Their paper was intended to be a follow-up to the Gibbs and Tucker model curriculum [1]. It discussed the need for faculty research laboratory facilities (not pertinent to this discussion) as well as laboratories to augment computer courses. Parker, Cupper, et al. defined an informal laboratory as one in which students work on self-directed assignments on their own (an open lab); a semiformal laboratory as one which is scheduled and during which an instructor is available for help; and a formal laboratory as one which is scheduled, attendance is required, all students work on the same product, and teaching staff is available to assist.

The advantage of a semi-formal over an informal laboratory is to lower the student frustration level and improve faculty efficiency since questions can be answered on the spot, thus avoiding long lines during office hours. A disadvantage is that only highly motivated students will take advantage of this opportunity. The others will procrastinate until the assignment is almost due. The scheduled and structured formal laboratories require "students [to] participate in the [lab] activity in a supportive and guided environment, rather than watching passively or struggling alone." [6, p. 213] This three-tiered laboratory organization breakdown concentrates on the function of a lab in the broader teaching/learning environment.

At a panel on issues in implementing structured laboratories for Computer Science at the 1992 SIGSCE Technical Symposium [7], Kenneth Abernathy and Thomas Allen from Furman University introduced still another set of lab organization terminology which emphasizes the problem solving process. They distinguished between non-structured (open) laboratories, where a programming assignment is the product, and structured laboratories where the emphasis is not on developing code but on processing the lab experience. At Furman, each structured laboratory must be written up by the student in a lab report format, much like those in the traditional science laboratory courses.

The open/closed laboratory organization terminology seems to be the most prevalent and it is these terms that are used in Computing Curricula 1991 [3]. "An open laboratory is an unsupervised assignment that may involve the use of a computer, software, or hardware for its completion. Students can complete an open laboratory at their own convenience. Conventional programming assignments are often done in an open laboratory setting.

A closed laboratory is a scheduled, structured, and supervised assignment that involves the use of computing hardware, software, or instrumentation for its completion. Students complete a closed laboratory by attending a scheduled session, usually 2-3 hours long, at a specific facility. Supervision is provided by the instructor or a qualified assistant who is familiar with the details of the assignment." [3, p. 19]

At the 1991 SIGSCE Technical Symposium, James Roberge' and Candice Suriano from Illinois Institute of Technology introduced the concept of the pre-lab, in-lab, and post-lab phases for closed laboratories [8]. Others, including Abernathy and Allen [7], have adopted this terminology in designing labs. Maharry and Singer [9] use the terms pre-conditions and post-conditions instead of pre-lab and post-lab. Roberge' and Suriano's emphasis was to help students become active and creative partners in the learning process. Their laboratories are based on the model used in engineering design courses rather than the model used in the physical sciences. The role of their labs was not to "supplement the ... usual programming ... assignments" [4] but to become the focal point of the course.

"The pre-lab phase is designed to immerse the student in the lecture material and encourage her to experiment with this material. Typical pre-lab exercises involve hand-tracing, developing a test plan, implementing an algorithm, ... These are to be completed before the lab.

"The in-lab phase uses the work the student has prepared during the pre-lab as the basis for a carefully channeled investigation of lecture concepts during the scheduled lab. Typical in-lab exercises involve tracing execution of a routine using a code visualizer, debugging a routine, adapting an algorithm to solve a well-defined problem, ...

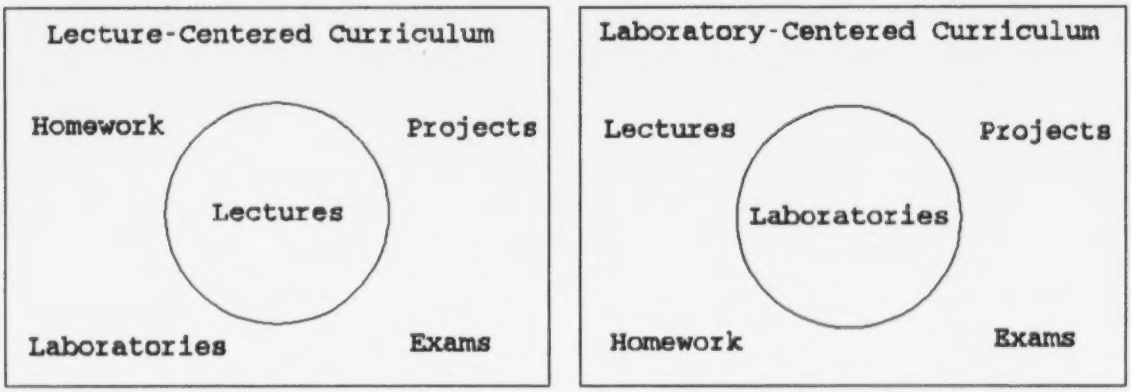
"The post-lab phase is intended to develop the student's ability to form and express analytical judgements. The student prepares a written analysis based upon some aspect of the laboratory material. Typical post-lab exercises involve comparing the performance of algorithms, proving an algorithm is correct,..."

At Saint Mary's, we have had a scheduled laboratory associated with our CS 1 course for nearly 15 years. From the beginning we have maintained an emphasis on designing a program before writing it. Hence up until the last few years, a student would prepare before or during lab a design for a laboratory programming assignment (these were in addition to the traditional homework programming assignments) and start coding the program. She would then finish coding and testing the laboratory program on her own in an open lab setting. Two years ago we extended the closed lab time from one to two hours and attempted to introduce laboratory assignments which encouraged students to experiment with computer science concepts while maintaining our traditional emphasis on design. I will talk about the details and partial results of this experiment later in the paper.

Development of Materials

When the push to introduce closed labs into the computer science curriculum began, many faculty were concerned about the shortage of materials available from which to design labs. Even the Computing Curricula 1991 report [3], which strongly encouraged the use of closed labs in all computer science courses, did little more than list sets of laboratory topics associated with the knowledge units.

At first, faculty used to the traditional lecture/assignment mode of teaching computer science saw the labs as an addendum to the course. The course would remain lecture-centered, but supplementary materials to solidify the lecture concepts would be presented and worked through during lab. The first place to look for laboratory materials was the publishers, and they responded with lab manual supplements for popular textbooks or lab manuals tied to particular versions of Pascal [10, 11]. These early manuals did provide practice with programming concepts introduced in lecture, but did not emphasize design nor encourage students to wrestle with the problem solving or non-programming aspects of computer science. I tried using ideas from [10] to design my closed laboratories for a couple of semesters after we extended the lab time from one to two hours, but found students doing the lab work by rote. The laboratories seemed to have little creativity, and by de-emphasizing design (I usually gave students the partial code and had them change or add to it) I found that students were less able to handle a programming project than before.



From notes prepared by Roberge' and Suriano for the 1992 SIGSCE Technical Symposium Tutorial "Using Laboratories in Your Course."

Motivated by innovative ideas from faculty like Roberge' and Suriano [8], Abernathy and Allen [7], and others, computer science teachers have begun to see CS-1 as laboratory centered rather than lecture centered. The slides above show that, in a lecture centered curriculum, homework prepares students for lecture and laboratories are used to solidify the material presented in lecture, whereas in a lab oriented curriculum the lecture prepares the student for laboratory work and homework is used to analyze the work done in the laboratory. Once faculty accept the lab oriented nature of CS 1 they will see the need to prepare laboratory assignments without relying on canned laboratories prepared by someone else. Most faculty do not use canned lectures; why should they use canned labs once the laboratory has become the focal point of the course.

In my experience, once the commitment is made to treat the laboratory as the center of learning, lab preparation, while time consuming, flows naturally as the syllabus evolves. The more lead time given to lab planning the better the course topics connect with one another. I found that programming assignments should also be considered when planning labs so modules needed in these assignments can be developed in lab.

Improved lab manuals [12, 13] have begun to appear. These place much more emphasis on the pre-lab and post-lab phases as well as providing more sophisticated and creative in-lab exercises. A group of faculty who attended the SIGSCE post conference workshop on Creating Structured Laboratories for Computer Science have set up a users group on the Internet for exchange of ideas and materials. The contact for this group is Debra Burton at the University of Texas at Austin (debra@cs.utexas.edu).

Role of Tutorials

Parker, Cupper, et al. [6] had a brief discussion of the benefits of a good tutorial to help integrate laboratories into the curriculum. They emphasized how important it is that tutorials be correct in every detail and robust enough so the student does not get lost and be unable to complete the tutorial. They found that tutorials worked best within a supervised laboratory session where help is available.

The circumstances which motivated me to develop on-line tutorials was the frustration of trying to teach syntax and programming procedures in the lab. If I tried to get the students to pay attention to me through all the distraction of a lab setting as I went through a lecture or demonstration in front of the room, they would not pick up enough to work on their own when I had finished. If I had them work along with me in a lock-step fashion on their computers, I was constantly moving from station to station trying to bring the students back to the right place after a bad keystroke. There was not time to teach syntax and how to work with Dr. Pascal (our

programming environment) in lecture if I wished to cover concepts more central to the core of computer science, but I had bad experience trying to lecture during the lab session.

Motivated by these constraints and the success I had experienced writing on-line tutorials for skill-oriented units like Word Perfect, Lotus, dBase, and Basic, I developed two tutorials for our CS-1 lab. The first leads the student through a first session with Dr. Pascal. The top part of the screen displays a model of the actual Dr. Pascal menus, editing, and execution trace screens, and I place information boxes at the bottom of the screen. In these boxes I ask students to do things like enter some code, save, load, compile, run, print, etc. I have complete control over the keyboard, so if they hit the wrong key, they get a helpful message about what they should have done and are given another chance to do it. With an on-line tutorial it is impossible for students to get lost. Sometimes they get frustrated because the on-line hints are not sufficient to help them find the right answer or an approach they try, while perfectly correct, is not the one the tutorial expects. But, generally, the students had a very positive response to this tutorial. The first laboratory session would start with the students running the first tutorial, and then they would be asked to use the actual Dr. Pascal package to enter and run a simple program.

Using a tutorial to introduce students to the use of a package and to the beginning syntax of a programming language has a number of advantages. Besides avoiding the lock-step presentation described above, it allows faculty and lab assistants to circulate among the students giving individual attention when needed. Students learn at different rates and a tutorial makes allowances for this difference. When a student makes a mistake in a tutorial, she gets instant feedback and a chance to do it right, thus avoiding the mystery of where and what the error is, so often experienced by many beginning programmers.

The second tutorial attempts to take the student through the design process. The main menus for the two tutorials are listed below. This tutorial was an ambitious project and will require revision before the next time it is used. Students found it helpful as a supplement for the text [14], which covered the design of programs very thoroughly. The main advantage of the tutorial is not in explaining how to design a problem solution, but in providing practice in actually working through a design. This laboratory was conducted during the second week of the course. After a student finished the tutorial, she was asked to design the solution to two problems, one using conditionals and the other using loops. This assignment required a lot of interaction with the lab assistants, but the tutorial at least helped most students know where to start.

Tutorial Menu Screens

First Tutorial	Second Tutorial
1. Introduction to Dr. Pascal	Review of Dr. Pascal
2. Setting up a program file	Retrieving a separately saved procedure
3. Saving a program	Data Objects, Values, and Types
4. Entering Procedures	Program, data, and display states
5. Compiling a program	Designing an algorithm
6. Running a program	Display, Read, and Assign actions
7. Correcting errors	Straight line sequence of actions
8. Printing program listing and output	Conditional choice of action
9. Exiting from Dr. Pascal	Repeated actions
Q. Exit from this menu	Exit from this menu

With another semester of supervising CS-1 closed labs completed, I have found other places where tutorials would be helpful: reading and writing to files, user-defined types, complex conditionals and case statements, mechanics of searching and sorting, working with strings, etc. I find it an enormous expenditure in time and energy to design and implement a tutorial -- up to 50 hours for a half hour tutorial. It is a little easier after creating the first tutorial for a particular application to do the rest since the work of designing the screens can be reused. It is important to briefly review past work in each subsequent tutorial and cover just enough for students to be able to complete the laboratory assignment. The temptation to have screen after screen of information without much interaction must be avoided. A handout can provide information. The role of a

tutorial is to provide practice in a controlled and supportive environment. It is important to help students be independent learners, but not frustrated learners. Tutorials are helpful in setting good habits, developing skills, and learning to use software tools.

I have developed the CS-1 tutorials using PC-PILOT in an authoring environment called PRO-PI. It is a PC-based package since the CS-1 labs are held in our PC student lab. PC-PILOT gives me control over every screen pixel and keystroke and will search student answers for keywords even if they are misspelled, so I can ask some open ended questions. PRO-PI makes it easy to organize the overall tutorial structure, call modules, and handle the menus. The learning curve is fairly steep, but I am pleased with PC-PILOT's ability to let me simulate any application. There are three graphics editors, the ability to capture screens, 16 colors in foreground or background, and the ability to create windows.

Saint Mary's CS-1 Lab Experience

As mentioned earlier, Saint Mary's has a long tradition of requiring closed labs in CS-1 and also emphasizing the design process with programming assignments. We have encouraged students to work on a design as a pre-lab assignment, but we are careful to check the design before the student begins the in-lab assignment. We found many of the suggested laboratory activities in [10, 11] frustrating since students were given a section of code to complete, modify, or expand without fully understanding the overall problem which gave rise to the code. As a result we have tended to interweave the concepts we wish the lab to emphasize into a complete problem which requires students to design at least some of its subprograms in order to implement it.

Some students have expressed frustration at how long it takes to do the labs, with much of the time spent typing, rather than thinking. Some are tempted to just copy their neighbor's code in order to get finished. As a result, we have often provided a complete design of the main procedure as well as the ability to download its code, so students can concentrate on subprograms which illustrate lecture concepts. Often in lecture we will develop general algorithms for concepts like searching or sorting, and ask students to implement specific applications of these concepts in the laboratory. We have incorporated labs on debugging and developing test data this semester and hope to allow more experimentation in our labs in the latter part of the semester after the design process has become second nature. The current set of laboratory assignments together with the results of the student evaluations is included as Appendix 1.

Conclusion

Teaching CS 1 as a lab centered course has greatly improved the lectures as well as the laboratories. Since implementation details are now covered in lab, we can develop the underlying concepts of the discipline in greater depth in the lecture. Our emphasis on design makes it possible to deal with lecture material on the algorithmic level, since students are very comfortable moving from algorithm to code on their own. The lecture covers material which gives background before the lab and further analyzes the results after the lab -- the real focus of learning is the laboratory.

To be successful, laboratory assignments must be well defined and have closure. Most students should finish the lab work in the time allotted. In our experience, it is important at the beginning of the course to have the students type in most of the lab programs because they need this experience to solidify the art of programming. In the middle and latter part of the course students should be able to download most of the code, coding only the parts pertinent to the concepts presented in that lab session. We found it difficult but very desirable to include experimentation in lab assignments. This helps challenge the good students while giving the weaker students a chance to finish the basic assignment.

Our experience has been that using tutorials is much preferable to lecturing about syntax or procedural concepts during the laboratory sessions. We found the idea of pre-lab and post-lab assignments helpful but only if there is a lead time of several days between the laboratory assignment and the lab session. We learned a lot by having the students fill in a short evaluation questionnaire during each lab session.

Developing a successful set of laboratories is difficult and a forum for exchange of ideas among faculty is essential. Ideally, the lab sessions should be outlined first, and then the lecture topics and programming assignments should be developed to ensure that the concepts of modularity and reusability emphasized in lab will be incorporated into lecture, homework and tests. A well prepared set of laboratories can generate a lot of enthusiasm and a real sense of accomplishment among beginning students and really revitalize the teaching and learning in CS 1.

Bibliography

[1] Gibbs, Norman E. and Tucker, Allen B. "A Model Curriculum for a Liberal Arts Degree in Computer Science." Communications of the ACM, 29(3):202-210, March, 1986.

[2] ACM Curriculum Committee on Computer Science. ACM Curriculum '78. ACM Curricula Recommendations for Computer Science, Volume 1. Association for Computing Machinery, 1983.

[3] ACM/IEEE-CS Joint Curriculum Task Force, Tucker, Allen B. (editor). Computing Curricula 1991. Association for Computing Machinery, 1991.

[4] ACM Task Force on the Core of Computer Science, Denning, Peter J. (chair). Computing as a Discipline. Association for Computing Machinery, 1988.

[5] Petersen, Charles G. and Miller, Nancy E. "Open Lab VS Closed Lab: Computer Programming I at Mississippi State University." SIGSCE Bulletin, 17(1):78-81, March 1985.

[6] Parker, Jeff, Cupper, Robert, Kelemen, Charles, Molnar, Dick and Scragg, Greg. "Laboratories in the Computer Science Curriculum." Computer Science Education, (1):205-221, 1990.

[7] Abernathy, Kenneth and Allen, J. Thomas. Creating Structured Laboratories for Computer Science: Tutorial. SIGSCE Technical Symposium, March, 1992.

[8] Roberge', James and Suriano, Candice. Using Laboratories in Your Courses: Tutorial. SIGSCE Technical Symposium, March 1992.

[9] Maharry, David and Singer, Carl. A Lab-Based Introduction to Programming. Presented at the Indiana Undergraduate CS Faculty Meeting, Franklin College, February, 1992.

[10] Dale, Nell. A Laboratory Course in Pascal. D. C. Heath and Company, Lexington, Massachusetts, 1990.

[11] Wilson, Carol W. Turbo Pascal Laboratory Manual. West Publishing Company, St. Paul, 1991.

[12] Decker, Rick and Hirshfield, Stuart. Lab Manual to accompany Pascal's Triangle Think Pascal 4.0 Version. Wadsworth Publishing Company, Belmont, California, 1992.

[13] Abernathy, Kenneth and Allen, J. Thomas. Experiments in Computing Laboratories for Introductory Computer Science in Turbo Pascal. Brooks/Cole Publishing Company, Pacific Grove, California, 1992.

[14] Pratt, Terrence W. Pascal A New Introduction To Computer Science. Prentice Hall, Englewood Cliffs, New Jersey, 1990.

Appendix 1
Spring 1992 Closed Laboratories for CS-1
Saint Mary's College
Lab Assignments Prepared by Mary Connolly

	Topic	Student Response *	Faculty Response
1.	Intro to Dr. Pascal Payroll Program	Tutorial helpful but experience is the best tutor.	Tutorial was effective and the payroll program was a good application.
2.	Intro to program design. Compute area and perimeter for carpeting a house.	Tutorial not as helpful as the first.	Tutorial repeated the info in the book. The carpeting assignment was hard - needed lots of individual help.
3.	Code and Run the designs from lab 2 using IFTHENELSE, WHILE, I/O	Much easier to code and run a program once the design is correct.	Reviewed the use of sentinel values from an example in the book. Good lab to see the complete programming process.
4.	Given the design, Code and run a simple IFTHEN test on test grades. Modify in stages to nested IFTHENELSE and sentinel loop.	Learned use of conditionals and defined constants. Dr. Pascal error messages were not helpful.	Hard to explain the benefits of nested IFTHENELSE over sequence of IF statements in lab.
5.	Use Data Files for input data. Use actual and formal parameters.	Use of parameters was confusing at first but became clearer as lab progressed.	Very helpful lab to understand I/O and parameters.
6.	Searching arrays of records. The designs of most of the four procedures were given.	Confused about parameters and subscripts. No problem with the concept of arrays of records.	Lab was too long. Should have had them download the main program and some of the procedures.
7.	Access data using functions, string variables and user-defined types. Had to design five procedures and one function.	Beginning to understand subprograms. Some frustration with not finishing.	Too long. Many did not finish. Maybe too many concepts for one lab.
8.	Mailmerge a form letter using string functions. Design insert and delete procedures.	Hard to remember what they learned before break.	Tried to review concepts at start of lab. Not very successful. Students needed a lot of individual assistance.
9.	Create test driver for simple inventory control program which students downloaded from the network. Lots of semantic errors.	Not as much typing - more time for thinking - liked it better. Hard to find errors in program they did not write.	Successful Lab, but needed practice of earlier labs to make it instructive.

10.	Download a menu driven program. Design and code six procedures. Use CASE statement and defined constants with a menu.	Too long unless they wrote subprogram design before lab. They were frustrated at waiting until their designs were checked before starting to code.	Too many subprograms. Could have learned as much with a shorter menu.
11.	Sorting and merging arrays of records. Students downloaded main program.	Liked this lab much better because they didn't have to do the "dirty work" of typing main algorithm.	A little short. Good experience with sorting and searching. The y see the value of reusing modules.
12.	Review of functions and CASE while computing averages from a 2 dimensional table of test grades.	Confused about rows vs columns. Program was downloaded. Finished early (one hour).	Should have included more experimentation, perhaps compare with parallel arrays.
13.	Compare efficiency of linear and binary searches on lists of random numbers.	Had trouble finding places to put counters to measure efficiency. Had trouble with simple arrays. Found the results interesting.	Too short. Should have had them develop control paths to see where to put the counters.

* Using an idea presented at the Indiana Undergraduate CS Faculty Meeting on February 8, 1992 at Franklin College, Indiana, we gave an evaluation form to each student at the end of each laboratory session. See Appendix 2.

Appendix 2

Please answer this brief questionnaire to help us improve your lab experience.

1. I found the tutorials in labs 1 and 2
   ___ Very Helpful
   ___ Helpful
   ___ Not very Helpful
   ___ Waste of time
   Comments:

2. I found the problem(s) to be solved in this lab
   ___ Very interesting
   ___ Interesting
   ___ Not very interesting
   ___ Busy work
   Comments:

3. I felt that I had prepared for this lab assignment
   ___ Very well
   ___ Well
   ___ Not very well
   ___ I was lost from the beginning
   Comments:

4. This lab's work tied into previous lab and class work
   ___ Very well
   ___ Well
   ___ Not very well
   ___ It seemed to come out of the blue
   Comments:

5. If you finished the lab give the approximate finish time;
   Otherwise what didn't you get finished in this lab?

6. What did you learn from this lab?

7. What went wrong in the course of the lab? (e.g. lost files, made errors, didn't understand the directions, etc.)

8. Other comments about the lab.

A Laboratory Component for CS1

Thomas J. Smith
Charles H. Stolze
Department of Mathematics and Computer Science
Manhattan College
Riverdale, NY 10471

Introduction

Computer Science educators have come to the realization that in many undergraduate computer science courses a laboratory component is a necessary requirement [1]. However, the format of the laboratory component is the object of much research and experimentation [2,3,4]. Existing models range from the traditional labs used in the natural sciences and engineering to newer kinds of labs which depart from that tradition. This paper reports on the use of a laboratory component in the CS1 course at Manhattan College.

Statement of the Problem

A primary motivation for the introduction of a laboratory component into the CS1 curriculum at our College was the need to find a way to help beginning students become more active in the writing of computer programs. Experience in teaching CS1 had led us to the belief that, although students are willing and able, they have difficulty "getting started" in computer programming. Some educators have attributed this inertia to the passive nature of today's college students; others suggest it is the nature of programming itself [5]. Whatever the reasons, it was clear to us that the traditional lecture method of teaching was not the best way to teach CS1.

In the fall of 1990 we decided to introduce a laboratory component into our CS1 curriculum. Once that decision was made we had to deal with a number of constraints. The first was the fact that our CS1 course was a three credit lecture course which met three hours a week. The climate on campus at that time was not conducive for us to try to expand either the number of credits or the number of hours allocated for the course.

The second constraint involved the hardware; our department does not have its own computer laboratory. The College, however, does have an adequate number of microcomputers which our majors share with other students. Also, our CS1 classes meet in a classroom that contains 23 networked IBM microcomputers. The class size is never greater than 23 and this guarantees that each student has access to his/her own computer while a class is in session. The students also have easy access to the use of a microcomputer outside of class.

Specification of the Laboratory Component

Under the constraints of our problem we decided to integrate a weekly laboratory into the CS1 curriculum by changing one of the three lecture hours into a laboratory session to be conducted by the instructor. The plan was to have the instructor lecture for two of the three hours each week and then assign a problem to be completed by the students in a one-hour closed-lab session. The lab assignment would be based on the material presented in the lectures and the instructor would be present at the lab session to work with students on a one-to-one basis.

Implementations of the Laboratory Component

At the time this paper was written, the laboratory component had been used in the Fall semesters of 1990 and 1991 and in the Spring semester of 1992.

Our primary goal has been to change passive students into students who are actively writing computer programs. We think the lab component has helped to accomplish this goal. There is no question that it is now easier to identify at an early time in the semester those students who are having trouble as well as those who are catching on. Because of this we have become strong advocates of the use of a lab component in both CS1 and CS2. However, there is much to learn and do.

A positive result from the labs is that the strong students prosper in this environment; they tend to be more independent and open to new challenges. Also, the lab sessions have made it easier for them to work together; we think this is healthy since it simulates the reality of the "real" world.

One difficulty we encountered is due to the fact that we now lecture only two hours a week. This has forced us to re-think how to present the material. For example, less time is spent explaining the syntax of a language and more time is spent describing the semantics of that language. This makes it imperative that the lab assignments be truly integrated into the curriculum and that they deal with questions which focus on what is currently being covered in the lectures. Coordinating the lectures with the lab assignments is a responsibility which requires extra time and planning from the instructor.

It would be helpful if there existed a bank of lab assignments which an instructor could use to select sample labs. This might be accomplished through the creation and publication of laboratory manuals to supplement CS1 and CS2 textbooks. This is being done with success in undergraduate mathematics education [6] and is just beginning with introductory computer science books.

A second problem with the new approach is finding an optimum way to help students who are having trouble learning how to program. A one hour lab session does not give the instructor much time to work with students on a one-to-one basis. Often the strong students will help the weaker ones during a lab session; this is a positive step but it is not a solution. The instructor needs more than an hour to spend with students in the lab. Also, a tenable class size must be found to avoid the potential burnout of instructors who must read and grade lab assignments in addition to their other academic duties.

A third problem is the type of lab session to conduct; should it be a closed-lab or an open-lab? A closed-lab is one in which the student completes an assignment during the session. An open-lab is one in which the student "drops in" at his/her convenience to work on a problem; in the open-lab there is usually a qualified lab assistant present to answer questions.

We decided to use closed-lab sessions. Rather quickly, however, we encountered the situation in which the strong students were able to complete an assignment on time while the weaker students were not able to finish. This produced a stressful environment for some; that was unacceptable to us since our objective was to have the students involved in an activity they looked forward to and enjoyed. We therefore made the lab half-closed; we told the students to try to finish during the session but if they could not finish then they could complete the assignment in their free time and hand it in before the next lab session. This was a good idea since, as the course progressed and the lab assignments became more complex, even the stronger students were unable to complete the assignments in one hour; we began to think that perhaps a two hour lab would be more appropriate for this course. One hour seemed to be too constraining for both the instructor and the students.

The lab assignments were usually given to the students at the lecture preceding the lab session. For example, if the lectures were presented on Tuesday and Thursday and if the lab session was held on Friday then the lab assignment would be given on Thursday. The assignment would be discussed briefly, often at the expense of some lecture time, to insure that the students understood the problem. We began to realize that two lecture hours a week were not sufficient; a more appropriate model seemed to be evolving, one which allowed for three one hour lectures and one two hour closed laboratory.

In an attempt to get help during the lab sessions we encouraged some upper level undergraduate honor students to volunteer to serve as lab assistants. Our reaction to this has been mixed; some students were excellent lab assistants, others were not so good.

Sample Laboratory Assignments

The lab assignments fall into two categories:

1. Assignments which implement the material currently being presented in the lectures.

2. Assignments which encourage the use of scientific experimentation.

Most of the assignments are in category 1. Positive results, however, have been reported from the use of assignments in category 2, [7].

A sample lab assignment from category 1 is the following:

Write a program which solves a quadratic equation. The input for the program are the values of the coefficients a, b, and c of the equation. We supply the students with several sets of input values to be used for testing the program in the laboratory.

This assignment requires the use of the IF-THEN-ELSE statement and the SQRT function. The input values we give the students for testing the program include the case when a = 0 and the case when the value of the discriminant will be negative. This serves as an excellent way to demonstrate what is meant by a robust program as well as what is meant by an exception and how it can be handled.

A sample of an assignment from category 2 is based on preliminary work done on sorting. During the lectures we develop the algorithms for Bubble Sort and Selection Sort. We also develop the code for implementing these algorithms. We then give the following lab assignment:

We give the students a program which generates an array L of N integers where N is the size of the array and its value is defined in the constant section of the program. Each integer in the array is randomly generated. The program copies L into two array variables L1 and L2. The program contains a procedure which sorts array L1 via Bubble Sort and a procedure which sorts array L2 via Selection Sort. Each procedure contains counters which count the number of comparisons and the number of moves. Each procedure prints on the screen the number of comparisons and moves computed.

We ask the students to run the program many times; each time with a different value of N. We ask them to find the value of N for which there is a definite difference in the run times for Bubble Sort and for Selection Sort. Eventually Bubble Sort takes longer and we want them to discover for what value of N that begins to happen.

We also ask the students to record the number of moves and comparisons computed each time the program is run. We ask them to investigate the hypothesis that there is a connection between the size N of the array and the numbers being recorded. We ask them to try to discover a formula for the number of moves as a function of N and a formula for the number of comparisons as a function of N. We ask them to do this for both Bubble Sort and Selection Sort and then write a lab report describing their findings.

We allow the students to do this lab assignment in teams of two or three students. Each team needs at least two members in order to estimate the run time for the procedures. This lab serves as a precursor to the theoretical analysis of algorithms which is done in CS2. It is also an excellent way to get the students to write and to make presentations to their peers.

Conclusions and Recommendations

We are convinced that a laboratory component should be mandatory for both CS1 and CS2. The model we have described in this paper is adequate but not highly recommended. We are currently writing a proposal to be presented to the Curriculum Committee at our College; the proposal calls for a laboratory component in both CS1 and CS2. The recommended format is three lecture hours per week plus a two-hour lab session conducted once a week. Based on our experience we think that this format will work best.

We recommend that individuals using labs in the introductory courses start sharing laboratory assignments with each other. We suggest that, if there are enough interested ASCUE members, then a network could be formed of people willing to share their lab experiences in CS1 and CS2.

References

[1] Tucker, A. (ed) "Computing Curricula '91: Report of the ACM/IEEE Joint Curriculum Task Force," ACM Press, 1991.

[2] Roberge', J., C. Suriano, "Embedding Laboratories within the Computer Science Curriculum," ACM SIGCSE Bulletin, Vol. 23(1), pp. 6-10 (1991).

[3] Epp, E. , "An Experimental Computer Science Laboratory, " ACM SIGCSE Bulletin, Vol. 23 (1), pp. 11-13 (1991).

[4] Joyce, D. , "A Virtual Lab To Accompany CS1 and CS2," ACM SIGCSE Bulletin, Vol. 22 (1), pp. 40-43 (1990).

[5] Perkins, D. N. , et al, "Conditions of Learning In Novice Programmers," Studying the Novice Programmer," Soloway, E.(ed), Lawrence Erlbaum Associates, Publishers, pp. 261-279(1989).

[6] Roberts, W., "Resource Collections for Calculus," Computer Algebra Systems in Education NEWSLETTER, Dept. of Mathematics, Colby College, Waterville, ME 04901, Number 10, May, 1991.

[7] Baldwin, D. , "Using Scientific Experiments in Early Computer Science Laboratories," ACM SIGCSE Bulletin, Vol. 24 (1), pp. 102-106 (1992).

The Changing Role of Computing in College Administration

John A. Anderson
Director of Information Services
Loras College
1450 Alta Vista
Dubuque, Iowa 52001

In 1984 John Naisbitt predicted in his book, Megatrends, that the world was in a state of drastic change. He felt that this phenomena could best be described by a few general trends:

1) the change from an industrial to an information society
2) from centralization to decentralization
3) forced technology to high tech/high touch
4) from a national economy to a world economy
5) emphasis on long term rather than short term
6) institutional help to self-help
7) representative democracy to participatory democracy
8) hierarchies to networks
9) regional shift of population from north to south
10) either/or decisions are becoming multiple option

Of these trends, many have since that time proven themselves to be very accurate predictions. They also do a very complete job of summarizing what is currently happening in higher education institutions and more specifically computer departments in college administration.

These general trends that have changed the way we look at the world have coupled with increasing financial restrictions associated with the current recessional economy to create the most difficult economic crisis in the history of our colleges and universities.

Problems Facing Institutions and Computing

The combination of rising operational costs and shrinking funds for purchasing adequate resources has made it necessary for computer operations to do more with less. These trends have even countered the fact that hardware costs are shrinking rather than rising. The money that might be saved on the hardware side is quickly consumed by the ever-growing expenditures for personnel, software, and communications. With networks and connectivity being the primary technical trend, today, it is evident that the key cost to computing will be in the communications arena.

On many campuses, the effects are evident in all aspects of their operations. Student services are being eliminated, course offerings are being diminished, campus up-keep is being delayed or forgotten, complete departments are being eliminated, and staff are being retired early, laid-off, or fired. In these tough times, the drastic budget cuts are especially felt in the areas of research and administrative computing. The last things that any college can do is to cut technological support for the academic sector, lower academic standards, or decrease financial assistance to its students. Because of the intense competition between colleges and universities, it is essential to demonstrate the ability of the institution to keep up with these changing times and not compromise their product. The most obvious result of this situation, however, is that tuition must be significantly increased, further jeopardizing the overall marketability of the institution.

Competition with other institutions of higher learning have forced college administrators to start thinking in terms of higher education as a business. This is due primarily to the shrinking numbers of college-eligible 18 year old high school students. When you add regional shifts in population to this formula, the result is very difficult recruiting for many colleges. This leads many schools to seek more information about what types of students they have, what is their competitive "edge", what do their students want, and how do they keep the students they have. These new types of information are being requested every day, increasing the workload of most administrative computing departments.

The rapid change in technology, the resistance to technological change, shrinking funds, and too many other important issues on campus, make life very difficult for the administrative computer support group. It is truly great what has happened in the advancement of hardware and software, but with the changes being made on a constant basis, everything is obsolete once it gets fully implemented. Technology changes so fast in the hardware arena that vendors can not even guarantee continued support for more than a year in the future without further upgrade and expenditure of funds. Software development also continues to lag behind the development of hardware, so that only some of the potential of the hardware can be fully realized immediately. Most vendors can supply the ordinary data processing needs for college administration, but now you must look much deeper at the special additional features of the software to find a more complete match for your organization. We currently continue to look for total solutions for our information needs of today, but also keep an eye on the flexibility of the software for future technological developments. If you do decide to purchase a packaged solution, you will look forward to performing endless testing and implementation of features that you don't want or need. Revisions seem constant and they never seem to arrive at the right time. This also adds to the over-crowded workload of administrative computing.

While technology is changing so rapidly, we still encounter continued resistance to these changes. It is understood that all change brings resistance, but the negative feelings about changes in technology still remain very strong. Sure, you have probably converted many people into believers, but there are still people that refuse to be automated. There are still administrators that feel that technological change is a threat to their position, an intrusion in their life, and a de-humanizing factor that sacrifices their individuality. Until computing, itself, becomes almost transparent and an everyday habit, people will continue to have this negative view of technology. Usually the root of this situation is a lack of education and a fear that they can't learn new methods and losing their job will be the result.

While the computer staff is struggling with these problems and trying to provide the necessary information support, college officials are busy dealing with even greater issues. Sky-rocketing alcohol and drug abuse; a climbing campus crime rate; and cultural, gender, and economic diversity of the student body and staff are all great issues that can cause a very negative view of how well our colleges are being run. These problems cause many administrations to divert a large amount of their attention to constant evaluation of their policies, functions, and priorities, leaving very little time to give any attention to the growing problem of meeting our information needs.

Major Issues in College Computing

As if these problems caused by government, economy, and culture are not enough, we must also deal with issues that have arisen due to the rapid advancement of technology on our campuses. Some of the most important issues are:

- separate or combined academic/administrative computing,
  academic access to administrative information,
  campus-wide networks and connection to outside world

- electronic connection of academic libraries for sharing
  resources and reducing requirements for physical
  space

- electronic data interchange (EDI) for transfer of official institutional documents
- touch-tone telephone registration
- software acquisition; purchased or home-grown, open systems or proprietary, purchase or lease, use of outside consulting, and use of CASE tools for software development
- security issues; legal, technical, and ethical aspects, departmental ownership of data, privacy laws, sharing security responsibility with users, and protection of data from users
- disaster/contingency planning, risks, and prevention
- training and re-engineering technical staff; professional development, career advancement, motivation, productivity, recruiting, retention, and evaluation
- distributed processing; networking, RDBMS, user control, technical coordination, client/server, and decentralization of computing

All of these issues are critical for the continued success of most collegiate administrative computing departments. This is why it is so important that administrators be forced to be aware of the opportunities that are available with their careful spending of scarce funds on technology.

Administrative Computing's New Role

All of the factors mentioned to this point are combined to force the administrative computing staff to adapt into a new role in the college administration. I believe that the computing staff will be without a position if they insist on only developing software and operating hardware. These jobs are being taken over by end-users as they increasingly gain more technical expertise. The computer support positions must be able to convert into the support that users now need. We will probably always have a few users that require the hand-holding that accompanied the "golden years" of computing, but the truth is that the new end-user no longer requires the companionship of their programmer to accomplish their computing goals. The computer staff, however, must assume some kind of responsibility for controlling technological development with a consulting role. If users are not writing their software, then vendors are. Either way, technical staff must still control, not the users or the vendors.

Administrative computing has taken on the tasks of assisting end-users with policy and procedure development, technical advisement, resource planning, facilitating access and connectivity, educating in technical skills, and general coordinating of security and control. Our image is considered more professional and, in most cases, computer staff are finally perceived as a major component of the overall administrative effort.

The old days of loading cards, mounting tapes and disks, and printing output have evolved into computer staff involvement in educating and coordinating other administrative personnel in making major institutional decisions. With end-users now developing their own applications and integration of data between offices being the basis of administrative computing, we have been forced into a role of motivation, support, and leadership. It has become essential for various departments of end-users to communicate on a regular basis, and the computer department plays a key role in coordinating this process. We have now taken on a "Peace Corps" attitude of teaching staff to "do-for-themselves" rather than the old approach of "do-for-the-user". The reactive "squeaky-wheel" methods of a few years ago have been discarded for a pro-active leadership role bonding the college administration in a single unit, all striving to reach the same objective. This new professional image has even paved the way for a new executive position in some administrations referred to as the "Chief Information Officer", a term that before this time had only been seen in the business sector.

With this new supportive role, come new responsibilities and expectations. As end-users learn to do more technical tasks, they require more knowledgeable support. This puts the computer staff in the unenviable position of improving their knowledge a few levels higher than their advancing end-users. The result is a position of consultant and advisor based on education and awareness. It is more important than ever for technical staff to fully understand the business of higher education and specifically the needs of their users. Obviously, the computer support personnel must become more people-oriented so they will understand the viewpoint of their users in dealing with hardware and software problems. This will result in the support staff being more sensitive to critical problems that occur due to a lack of information or technical performance.

Relationships with vendors, users, faculty, and the public have had a drastic effect on how technical staff deal with their new responsibilities. The relationship between technical personnel and hardware/software vendors has developed into a partnership with the result being a workable solution for the administration of the institution. An effective vendor will seek out the ever-changing needs of the customer and align their products to provide complete solutions. As packaged solutions are purchased, the computer staff again relinquishes some control to the vendor, forcing themselves to weaken their position in software development, and strengthening their role in support and control. Because the vendor almost becomes part of the administration of your institution, it is important for them to be well in-tune with your mission, desires, and priorities. The computer staff must become a liaison between the vendors and the end-users, coordinating meetings between the two entities for an improved flow of information. Vendors must know their customers and customers must trust and be comfortable with their vendors for the relationship to continue to be a positive one for the institution.

Our relationship with end-users has definitely been the biggest change in administrative computing. The technical staff have also developed partnerships with user departments as they have become more knowledgeable about their computer use. It used to be that unless an office was using an established (proven) computer procedure to perform day-to-day processing, computer staff needed to beg and force administrators to use the system. Today, we look for alternate methods of computing to control the workflow on the main computer systems. With the growth of distributed processing, we see the users continually taking on new experiences and being successful at them. It is obvious that, we, as computer professionals have seriously under-estimated our non-technical users for quite some time. As users learn the computer jargon and build their confidence in technical areas, we see a much better two-way communication between the groups. The bottom line is a greater understanding of end-users by the technical staff and increased user involvement in software and hardware support. This provides the computer staff with more information than they have ever experienced in the past. For once in computing history it seems possible that technical staff and end-users may be able to work together in a team concept.

Maybe a little more subtle is the improving relationship being enjoyed between technical staff and teaching faculty at a number of institutions. We have reached a new level of understanding between these two groups, where there was little tolerance in the past. Faculty had a hard time grasping the concept that academic and administrative computing had many similar but also very different needs. The bonding agent in this relationship, today, is definitely the data that the computer staff controls and the realization that this data is a significant resource for faculty as well as the rest of the institution.

Along with all these other relationships, the coordination between institutions and their local communities has grown immeasurably. This relationship has become evident in many computing departments by an increase in requests to perform tasks for off-campus entities. As the rest of the world becomes aware of the significant role that colleges play in their communities, it becomes more important for colleges to determine what value it brings to the region, how the college is seen by off-campus observers, what functions can be provided to the community, and what kind of leadership role the college plays in critical issues facing the community. As we accept this community responsibility, computing staffs are relied on to provide information on image, public-relations, worth to the area, and generally providing the resources necessary to perform business in the community in an efficient and professional manner.

To become the foundation of all these new developments, computer directors must concern themselves, more than ever, with the difficult task of maintaining an adequate staff to fulfill all the emerging expectations.

There are a number of factors to be considered when acquiring new staff or working toward professional development of existing staff. It is important to find personnel that are receptive to new ideas, will utilize new technology, possess a broad knowledge of business and computing, and have excellent public-relations skills. The computer person of today must be able to communicate effectively and above all else be an effective listener. General knowledge and flexibility have taken the place of technical specialties and other unique computing talents. You must be able to find people that seek more responsibility, desire to develop their careers, and want to continually upgrade their skills. The key to success is to be able to apply technical knowledge to business problems. This is not a given, since this takes a special blend of creativity, experience, and skill.

Reasons for the Changing Role

I have already mentioned a number of factors in the institution and the world, in general, that have forced administrative computing to change the way they look at themselves and how others perceive them. I feel that there are some other major factors that contributed to this development:

- more attention given to accountability
- increasing change to decentralized computing
- promotion of the concept that computer staff are professional and very necessary
- increased importance of efficiency
- concern for secure, reliable, timely, accurate, shared, and understandable data
- past successes in providing computer services
- better knowledge of end-user needs and concerns
- increased user expectations and demands for support
- better understanding of administrative goals by faculty, administrative end-users, the public, vendors, and the technical staff
- merging of all technologies (voice, data, video)
- increased complexity, scarcer resources, and more rapid change and growth

I don't believe any single reason is more responsible than any other for the significant changes we have seen in administrative computing. I think that these changes have been happening quietly for a number of years and have only recently become obvious due to the tough times that colleges are currently experiencing. There are, I am sure, a number of other factors that were just as important as these, that I have not even noticed.

Undoubtedly, the biggest change in the institutional philosophy on technology has been the recognition that data, information, and the resulting knowledge are an extremely valuable institutional asset. Tying in with this revelation is the death of the traditional concept that academic computing only deals with classroom instruction and administrative computing is only concerned with accounting, admissions applications, student records, and fund raising. Only in the last few years have college executives begun to accept computer technology as an effective tool along with pencils, pens, paper, and adding machines.

In dealing with computing in the last few years, some of the personal observations I have made are that tradition is slowly being replaced with efficiency, there is a general trend to de-emphasize the idea that information technology is responsible for the critical problems in higher education, and that administration finally believes that technological change can be predicted, managed, and controlled in the long run with proper planning. Through experience, I believe colleges have learned that data and technology must be managed and that it is not in the best interest of their institutions to allow haphazard technological development on their campuses.

New Information Demands

With increased awareness of the capabilities of administrative computing centers, come new requests for information from all directions. The simple task of data extraction and reporting have become end-user tasks, but the added need for statistical information and more complex data analysis have required new types of computer support. Simple data processing has evolved, along with the changing needs, to become management information systems. MIS, with additional information requirements, has become decision support systems (DSS). Further information demands have forced the development of executive support systems (ESS) and more specifically executive information systems (EIS). The executives have finally decided that they can play an active role in computing without appearing like a technician. They are learning that as software systems improve their functionality and performance, that they don't need to delegate all information needs to subordinates. Usually more accurate, secure, and reliable information can be retrieved on the executive's desk rather than channeling through many other offices.

The most important aspects of providing this new information is that everything needs to be based on the most reliable information and access must be simple. To improve access to data, we have been utilizing the link between main systems and personal computers as much as possible. This allows us through terminal emulation and file transfer, to off-load many functions that are more efficiently performed on PC spreadsheet, database, and word processing software.

In data processing and MIS support, we usually were only concerned about recording transactions that had already occurred and then summarized and reported the results. With decision support systems, we were requested to perform some level of prediction to support decisions made by middle management. Usually limited access and problems with adequate data interpretation characterized early decision support systems.

Executive information systems are usually the most critical portion of an executive support system, that typically also includes some form of modeling and office automation software. A good EIS package is user-friendly, fast, easy to learn, flexible, and expensive. The package usually has great graphics capability, real data is not made available, and there is some control of interpretation. It is very important that the data is presented and interpreted in the correct manner. The system must rely on database and communication information both internal and external to the institution. The statistical analysis of data is an integral part of the overall picture allowing for comparisons over time and with data from other similar higher education institutions to demonstrate trends. Success still relies on the fact that data is reliable and easily accessible. An EIS must, above all other functions, absorb the over-abundance of data that exists in administrative systems, today.

A large source for new information requests is the growth of strategic planning on our campuses. The target of strategic planning is to evaluate our strengths, weaknesses, and problems on a long-term basis to better position our institution for the continuous changes in the higher education market for the next five or more years. We seek to find a good balance between technology, needs, and planning. This planning must be a continuous re-evaluation of functions and priorities based on the institutional mission, a linkage of technology with people and processes, a long-range plan to guide staff to educate one another about each other's needs, an analysis of market trends and our institution's position in that market, a separation of real needs from all perceived needs, and a time-line for accomplishing what needs to be done.

The executive's need for information for planning has seen the greatest increase of all types of new requests in the last few years. The need for new information, however, is not only seen on the executive level. Many middle managers and end-users also are requesting information that is not simply extracted from institutional data and clearly presented. Some of these needs are listed below:

- information to solve limitations of physical space on campus
- interrogation and control of all correspondence
- supplying administrative information for classroom instruction

BEST COPY AVAILABLE

- information on quality and quantity of recruited prospects and current student body
- analysis of market competition and projection of expected enrollment
- analysis of student performance information for student retention and evaluation of institutional academic standards
- coordination of admissions recruiting and available financial assistance
- cultural, economic, and gender diversity information
- analysis of major prospective donors gift history
- tracking progress of crime on campus and academic progress of athletes
- analyzing faculty/administration turnover
- institutional research and measurement
- research on non-traditional students and their needs

There are many more than these, but I believe this list contains the requests that we see most often. The largest increase in requests have been in the area of statistical analysis. We typically perform some type of data analysis on our internal databases and also on opinion information from surveys created on our campus. A good statistical package has become an essential part of administrative information services. The most important factor to remember, is that you can perform the most sophisticated statistical procedure to analyze your survey results or database information, but unless it is fully understood, it is worthless. I feel that an improvement that will likely have an effect on this problem is the emphasis on presentation graphics with most statistical packages.

Another common request is the surveys from off-campus that seem to arrive faster than you can fill them out. Talk about information pollution! It appears that everyone is busy researching everyone else. The size of these questionnaires have gone from your standard one pager to 20 page booklets, produced by companies and organizations that nobody has every really heard of. This also seems to be how mailing lists are born. As a professional, however, you must be able to weed through these and fill out the questionnaires that seem the most legitimate. It is a very good idea to keep copies of responses to know what to respond on other surveys with similar questions. Keeping a history of responses, also helps you to monitor how much things are changing in your own institution.

Attempting to Meet the Challenge

I feel that the most important factor in fulfilling our new responsibilities is the establishment and maintenance of an efficient and effective technical staff for administrative computing. As the chief information officer (whether formally or informally), you must provide the following guidance for your staff:

- insure they know about business of higher education as well as technology
- motivate them to your satisfaction level
- keep in mind that they don't work for you, they work for themselves
- always set a positive, professional attitude
- stress self-evaluation
- know ALL talents and abilities of your staff
- always give them a voice and listen to them
- establish ownership and pride in what they do
- educate to achieve desired level of productivity

- communicate, tell all, be honest, and fill in the blanks so they won't
- reward major accomplishments
- avoid any confusion about responsibilities
- if there are problems, act don't react
- rotate areas of responsibility to promote creativity and enhance service
- be sensitive to institutional politics
- fight for professional development and advancement for your staff

Establishing and maintaining your technical staff is likely to be the most exciting challenge facing you in the next decade. There are, however, many other items that computer directors will need to acknowledge and be able to handle in the next few years. Among them are:

- support and encourage policies and procedures that will benefit information services
- establish a clear position of computing direction and how it relates to the college mission and live by it
- learn as much as possible about new technology
- continue to educate administration to allow informed decisions and establish a true understanding and appreciation for the value of good information
- spend as much time on planning as transaction processing
- turn visions into realities
- determine how much data should be retained on your system and weigh the risks of deletion
- distinguish between real and perceived needs and required and desired service levels
- insist on the involvement and committment of all levels within the administration
- continue to emphasize standards and documentation
- wise allocation of scarce resources with very little room for mistakes
- utilize the super-end-user as a consulting expert
- create a professional image of stability and expertise
- last, but not least, data must be valid, accurate, relevant, understandable, timely, and reliable

Technology and institutions of higher learning will never stop changing, so as administrators, we must adapt to the changes and learn to manage the progress of technology in realistic dosages for our user's consumption. The key to our success is that we continue to remind our institutions that good information does not just happen, it must be organized and managed to become useful knowledge.

Learning to Collaborate

William Doemel
Director, Computer Services
Wabash College
Crawfordsville, IN
Doemelb@Wabash

Introduction

As a microbial ecologist who spent most of his career studying extreme environments, I have been fascinated by the diversity of ways that members of different species actively and passively cooperate and collaborate to assure their mutual survival. For example, in the hot spring effluents of thermal springs, there are oxygen producing photosynthetic bacteria that exist on top of mats of anaerobic bacteria. The photosynthetic bacteria provide fixed carbon for the anaerobic bacteria while the bacteria provide a substratum for the algae. Existing in these mats are other bacteria that scavenge oxygen, keeping the level low enough that t he anaerobic bacteria can survive. For these organisms, cooperation is a necessity of survival for if you restrict light from the system, both the algae and the bacteria are washed away. If you limit the growth of the anaerobic bacteria, the algae disappear. And, while not demonstrated, I suspect that if you limited the growth of the oxygen scavengers, then the other organisms would suffer, as well.

Now, as Director of Computer Services at a small liberal arts college, I believe that cooperation and collaboration with other institutions is essential, if we are to assure that our students obtain the benefits promised by computing. I offer three experiences for consideration. I have chosen only to summarize the experiences at this point and not to offer conclusions. Those will have to wait for the talk and the ensuing discussion.

An Educational and Research Network for Indiana

Background

December, 1990.	Directors of computing from most of the higher education institutions agree to a feasibility study for a statewide data network.
January, 1991.	Two committees, Data Network Planning Committee and the D ata Network Governance Committee, started their deliberations.
May, 1991.	The directors unanimously accepted the recommendation of the Committee for the Indiana Higher Education Telecommunication System (IHETS) to d esign, fund, and manage an interinstitutional statewide data network.
October, 1991.	The Data Network Committee was constituted within the str ucture of IHETS. This Committee included representatives from the state universi ties, the state technical college, Notre Dame and from the private liberal-arts colleges.
December, 1991.	Grant requests submitted to NSF and Lilly Endowment for initial funding of the network. 32 institutions supported the applications.
March, 1992.	NSF approved the grant request.
May, 1992.	The first link of the network between Indiana University, IHE TS and the University of Indianapolis was installed.

Observations

Formed in 1967 by action of the Indiana General Assembly, IHETS serves higher education in Indiana by providing telephone and telecommunication links between in stitutions. While all institutions use the telephone services offered by IHETS, the state universities and the vocational college are the primary users of the telecommunications services. Governance of IHETS is through a hierarchical series of committees that report to a board consisting of eight institutional members, 7 of the 8 being from state-supported institutions. In large part, the governance and priorities of IHETS have been set by the telecommunications users, ie. the state universities.

While all but one of the universities are connected to the Internet, only three of the private institutions have Internet connections. Indeed, Indiana University operates a mini-statewide data network connecting all of its branch campuses and IUPUI. Purdue, while not having links to its branch campuses, has several connections to the Internet. So, contrary to the existing offerings of IHETS, the smaller institutions would appear to have more to gain by the establishment of a statewide, data network. Yet, the expertise for designing and maintaining such a network resides in the state institutions.

At first glance, Indiana University, Purdue University, IUPUI and Notre Dame, se em to have little to gain by participating in a cooperative venture to develop a statewide data network. Indeed, each might loose some control and perhaps some dollars. Should the network become a reality and follow the model of the other functions of IHETS, the state may assume a significant portion of the support of the network, perhaps siphoning off dollars that would otherwise go to the state institutions. Nevertheless, each of these institutions has expressed strong support for a state-wide data network and has contributed hundreds of hours of staff time. Why?

For their part, the private institutions value their independence. Indeed, outside observers might conclude that their only agreement is to compete for students and for recognition. With tight budgets, small staffs and few experts, they would seem to have little to offer to gain access to a statewide data network. Nevertheless, more than 30% of the members of the working groups of the Data Committ ee come from these institutions. Why?

The Monon Project

Background

Summer, 1990.	Discussions concerning a shared library automation system between Wabash and Depauw.
October, 1990.	Idea of a data link between Wabash and DePauw, the Monon Line.
Spring, 1991.	Hypercard Workshops between the two campuses.
Fall, 1991.	Feasibility project.
Spring, 1992.	Joint grant proposal to NCR Corporation for network hardware.

Observations

Wabash College (Crawfordsville, IN), and DePauw University (Greencastle, IN) lie 30 miles apart in central Indiana, connected by the old Monon Railroad line which used to transport shoppers and visitors between the cities. The two colleges have been rivals for over 150 years. That rivalry began on the athletic fields, but also now extends to recruiting new students, hiring new faculty, and providing outstanding facilities for learning. Both institutions are highly regarded undergraduate, liberal arts institutions. Both graduate students with exceptional records of professional and graduate school attendance. Both have provided chief executive officers and

presidents for multinational corporations. Similarly, both institutions are facing challenges. Wabash and DePauw have demographic predictions of shrinking pools of high school students from which to recruit new classes. Both face rising financial aid costs which will produce major budget challenges during the coming decade. At the same time, both institutions are faced with an information explosion most easily managed by new computer technology. Future language classrooms will include computers on which students will talk directly with native speakers. Biology students will be able to dissect frogs many times over through simulations which will allow for repetition and mist akes in the way that old dissection laboratories could never allow. Students not yet born during the Civil Rights movement will learn both rhetoric and history by reading and hearing Martin Luther King's "Letter from the Birmingham City Jail," gaining through technology the context of the times, the knowledge of the la ws surrounding King's arrest, and through QuickTime video, seeing his arrest as if they were there.

Faculty at both institutions are eager to incorporate these teaching techniques into their courses. But development of these course materials will require improved hardware and software at both institutions, networking capability so that faculty can work together and students can share the results, and staff who can work with faculty members to help them accomplish their academic goals. The cost f or both institutions to undertake these important improvements are prohibitive, considering current enrollment and budget challenges, if undertaken by either in stitution alone.

Together, however, the institutions could accomplish these goals. How? By sharing staff, by sharing equipment, and by sharing course development responsibilities between the two institutions.

Conclusions

I had hoped to be writing this paper with the certain knowledge that we had rece ived funding for our project and that the Monon "railroad" was in construction. We haven't and it isn't. Nevertheless, we have learned much from this collaborat ion. On small scales, we are sharing the responsibilities of educating ourselves , our staffs and our faculties.

Private Academic Library of Indiana (PALNI)

Background

State University Library Automation Network. Encouraged by a grant from Lilly En dowment, the libraries of the state universities formed a consortium to enhance access to their holdings. All agreed to adopt NOTIS as the software for their li braries and to initiate connections between their libraries.
May, 1990. Lilly sends an RFP to the Presidents of 30 Indiana private colleges a nd seminaries offering them the opportunity to obtain Lilly funds to assist thei r joining SULAN.

February, 1991.	Librarians from 25 of the private colleges form a consortium, in itially called IPCSL, now the Private Academic Library Network of Indiana (PALNI).
August, 1991.	Librarians decide to seek support from Lilly for a centralized, shared library system.
September, 1991.	Computer Center Directors urge librarians to consider an "open" library system utilizing TCP/IP protocol and the IHETS data network.
February, 1992.	Librarians select DRA as the library system and agree to an "open system" utilizing TCP/IP protocols over the IHETS Data Network.

Observations

With Lilly Endowment holding out the carrot of funds for library automation if the librarians would agree to select a system compatible with NOTIS, the librarians from the private colleges starting talking about automating their libraries. They determined that establishing a single system for each library would cost between $120,000 and $148,300/month after an initial start-up cost of between 5 an d 9 million dollars. On the other hand, a centralized, shared system would cost $63,000 to $72,000 monthly after an initial start-up cost of between 3 and 4 million dollars.

Their consultant urged them to adopt a "private" network with dedicated 19.2 KB lines extending between campus terminals and the main system. Such a system assured the performance of the library system, but limited access. With the support of faculty and administration, the computer center directors urged the librarians to adopt an open system utilizing the IHETS data network. While the performance of such a network could not be assured, users within the institution would have access to the system through their campus network. For most institutions, the aggregate cost of supporting both a library network and a connection to the Internet were too great for the campus budgets to absorb.

Questions

What do individual institutions relinquish when they enter into a collaboration with another institution?

Why should universities be interested in helping private institutions gain access to computing resources?

Don't you spend a lot of "wasted" time in meetings seeking compromises?

Knowing that you could gain support from an agency alone, why do you benefit by collaborating with another institution?

Computerized Academic Advising Package

Valerian S. M. John
Department of Mathematics, Computer Science and Engineering
Gurbax Singh
Department of Natural Sciences
University of Maryland Eastern Shore
Princess Anne, Maryland 21853-1299

ABSTRACT

This paper describes a software package developed for use by college and university faculty involved in academic advising of undergraduate or graduate students. Information on the courses required for graduation, courses completed, grades earned, etc., may be stored in the database for each student. This information may be retrieved, edited, updated at any time, and various types of reports may be displayed or printed. Thus, the advisor is able to track, assess, and report on the student's academic progress towards graduation with minimal effort. The package is written in dBASE III PLUS for IBM and compatible microcomputers.

Key words : Academic, Advising, Faculty, Microcomputer, Software, Student.

INTRODUCTION

Many of the faculty members at any college or university are required to advise students on their academic performance and progress. This task involves much paper work, time, and effort.

From the time a student is assigned to a faculty advisor, the advisor is required to maintain and update relevant information. This process of record-keeping may begin with the manual creation of a file, one for every student who is being advised. This file is created by collecting personal data like the social security number (SSN), name, major, minor, classification (freshman, sophomore, etc.), type (teaching, non-teaching, etc.); the courses completed thus far; grades earned; and courses to be taken in the next semester or courses to be completed before graduation. In addition, the advisor has to make sure that the student takes the required courses from designated areas, such as general university requirements, electives, supporting courses, and courses from the major area of concentration. He must also deal with situations in which the student registers for the advised courses but does not make acceptable grades, or drops out of some of them. Most of this information has to be updated as the student progresses towards graduation.

Using the information on file, the advisor is able to steer the student through courses which are to be completed according to the graduation requirements of the college or university. For this purpose, the advisor will refer to the course catalog, to access information on the courses required for graduation in each area of the curriculum; to look up equivalent or alternate courses in each area; to determine and compute the number of credits required and credits earned; etc. The student's success or failure to meet the grade requirements in each of the courses is also to be taken into consideration. This laborious task multiplies with the number of advisees a faculty member is assigned.

The use of a microcomputer can simplify the entire process; reduce needless thumbing through the catalog many times during the tenure of each student and paper work; and save time and effort. This paper describes a "**Computerized Academic Advising Package**" **(CAAP)**, developed for this purpose.

OVERVIEW

CAAP is a menu-driven software package written in dBASE III PLUS [1]. It is an integrated package comprising fourteen programs and is executable on IBM PC, PS/2 and compatible microcomputers under IBM PC-DOS or Microsoft DOS. Besides the menu handlers, there are database management programs to create, edit, store, and retrieve information from the Personal Database, Requirements Database, Completed Courses Database, and Courses/Credits Database. Report generation programs are provided for printing a 'Program Evaluation Sheet'(PES) or a comprehensive list of courses offered. The PES is used at the University of Maryland Eastern Shore (UMES) as the final report on the student's completion of all requirements. This report is identical to the existing document manually-prepared by the faculty advisor.

DESCRIPTION

The main menu of CAAP shown below appears after the start-up screen.

```
MAIN MENU                          UMES - CAAP 1.0

        1 PERSONAL INFORMATION

        2 GRADUATION REQUIREMENTS

        3 COMPLETED COURSES

        4 COURSES & CREDITS INFORMATION

        5 REPORT GENERATION


        0 EXIT TO DOS
```

This menu leads to five sub-menus providing services for: personal information maintenance, graduation requirements entry, status of courses completed, courses/credits database maintenance, and report generation. Any of the services provided by a menu may be selected either by moving the highlighted bar using the up/down cursor control keys and striking RETURN/ENTER or by pressing the appropriate option number. Pressing ESC at any menu takes the user back to the previous screen, i.e. the main menu or DOS depending on the menu where ESC is pressed.

Personal Database Maintenance

Each student being advised is registered with CAAP by creating a new personal information record in its database. Option 1 of the main menu allows the advisor to add a new student to the database by entering the student's SSN, name, classification (F-Freshman, S-Sophomore, JR-Junior, SR-Senior), major, minor, type (T-Teaching, N-Non-teaching, H-Honors); department; and names of the faculty advisor and the department chairperson. A typical data-entry screen for Personal Database maintenance is shown below.

```
PERSONAL INFORMATION                    UMES - CAAP 1.0

      SSN              123-45-6789
      LAST NAME        DOE
      FIRST NAME       JOHN
      MIDDLE INITIAL   I
      CLASSIFICATION   SR
      MAJOR            BIOLOGY
      MINOR
      TYPE             H

      DEPARTMENT       NATURAL SCIENCES
      CHAIRPERSON      JACK WHITE, Ph. D.
      ADVISOR          ROBERT SMITH, Ph. D.
```

Ret/Beg/End/Next/Prev/Skip/Edit/Add/Copy/Del/List/Filt/Tally/Help/Quit?

A variety of options are available as displayed at the bottom of this screen. Brief description of these options is given below. The same options are also available for the other databases.

Add	Add a record to the database
Beg	Go to the first record
Copy	Duplicate the current record
Del	Delete the record on display
Edit	Edit the current record
End	Go to the last record
Filt	set a filter using any matching field values
Help	Brief explanation of these options
List	Display records beginning from the current record
Next	Go to the next record
Prev	Go to the previous record
Quit	Terminate current maintenance level
Ret	Retrieve a record by key
Skip	Go forward/backward by a number of records
Tally	Count the records

Graduation Requirements Database

The advisor must maintain a record of the course codes (course numbers) and acceptable grades for graduation, for every major course of study, in the Requirements Database. At UMES, these courses are grouped according to the General Education Requirements (GER) under Curriculum Areas (I - V), Major Courses, Major Electives, Free Electives, and Supportive Courses. The options available for the manipulation

of this information are similar to those provided for the other databases. Depending on the student's major, the required courses are copied from this database into Completed Courses Database, which is explained below. For this reason this database must be updated at the outset before modifying the Completed Courses Database.

Completed Courses Database

This database maintains the codes of the courses required for graduation, and their corresponding semester credits; and courses completed and grades earned by the student. The required courses, and acceptable grades are looked up from the Requirements Database and are grouped under the different areas of the curriculum. The number of credits earned by completing these courses are looked up from the Courses/Credits Database and displayed on the screen. The advisor has to simply indicate if a course has been completed or not. If a course is completed, the grade and number of credits earned must be entered to verify against the requirements.

Courses/Credits Database

The Course/Credits Database is used for maintaining the titles of the courses offered by the University that are relevant to the major of the student being advised; course codes; and semester credits of the courses. Since the course codes are looked up from this database, it must be updated before entering data into the Completed Courses Database. A complete listing of these courses may be viewed on the screen or printed on a printer.

Report Generation

Option 5 of the main menu may be used to produce two types of reports. A PES may be produced for each student, summarizing the student's current status with regards to graduation. The PES is in the form of a letter from the department chairperson to the student. Along with the student's personal information, an area-wise listing of required courses and credits; courses completed and credits; and courses to be taken is provided. Thus the PES helps to point out the deficiencies, and the student can be advised to complete those courses.

A Summary Report provides a listing, for all the students in the database, of the total credits required and credits earned. This listing is an at-a-glance progress report on the students under the purview of the advisor.

SYSTEM DESIGN

The system is designed with three primary objectives, namely, entry/input of data, storage of data, and retrieval/output of the stored data.

Various screens are designed for the input of information. Genifer [2] is used to create the programs for the display of screens. The layout of the screen; and the title, type, and length of the fields, are defined. The entry and editing of the data is facilitated by the options listed earlier.

The data input through the data-entry screens are stored in the various databases discussed earlier. By defining their structures, these databases are created in dBASE III PLUS. The Personal and Completed Courses Databases are indexed on the SSN, whereas the Requirements Database is indexed on the major, and the Courses/Credits Database on the course code.

The layout of the reports generated by CAAP are created by using Genifer and dBASE III PLUS.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the constructive feedback provided by the faculty of the Department of Natural Sciences at UMES.

REFERENCES

[1] Using dBASE III PLUS 1.1, Ashton-Tate, 1986.
[2] Genifer 2.0 Reference Manual, Bytel Corporation, 1988.
[3] Programming with dBASE III PLUS 1.1, Ashton-Tate, 1986.

Safe Computing: Protection Against Viruses

Dr. Larry D. Jones
Dean & Asst. Professor, Computer Technology
Kent State University - Geauga Campus
14111 Claridon-Troy Rd.
Burton, OH 44021

Did you celebrate Michelangelo's birthday on March 6th this year? If not, did your computer have a party anyway, but you didn't like the result? That was the day a virus named after the famous artist of the 15th and 16th centuries was triggered to strike. Unlike some relatively benign viruses that only flash messages like "Legalize Marijuana" on the screen, this one was more malevolent. It's mission was to overwrite the data on your hard drive's boot sector, thereby destroying access. Fortunately, because of a great deal of coverage in the media, most PC users avoided catastrophe. However, this is only the beginning; viruses are here to stay. "No PC is immune from a virus. Unless you're willing to disconnect your floppy drives, remove your network card, unplug your modem, and disable all communication ports, your system is at risk...It's virus season all year round" (Taylor, 1992, p. 122).

What is a Computer Virus?

A computer virus is, fundamentally, a program created to infect other programs and/or disks. Like their biological cousins, they are often very small, with as few as a couple dozen lines of instructions and as little as 200 bits. What makes viruses different than other programs is their ability, mission in fact, to replicate themselves and send those offspring off to infect (Burger, 1991). Those clones might be transmitted via floppy disks, network cable, or phone lines. They tend to be written for specific operating system environments (e.g. IBM PC, MacIntosh, Amiga, UNIX), although a few have been able to cross over between two platforms. Most of the viruses that concern users, of course, are in the MS/PC-DOS based systems. Another interesting feature of viruses is that they can be polymorphic, changing their effect from time to time and system to system (Denning, 1990). Clever little buggers! Why do they exist? According to McAfee (1989, p. vii), "the virus problem is like the graffiti problem in metropolitan cities. A curious thing about vandals is that they revel in chaos."

A Brief History

Perhaps the first clue of computer programs with virus-like behavior came from John von Neumann, renowned mathematician in the 1940s. He suggested that software could make machines mimic animal physiology and behavior by adapting to the environment and changing in order to survive. In one of my favorite science fiction stories, Arthur C. Clarke (1968) seemed to follow von Neumann's notion in his creation of HAL, Heuristically programmed ALgorithmic computer. The software that gave HAL intelligence prompted him (it had a masculine voice) to investigate the unknown and adapt according to discovered information. Unfortunately, HAL seemed to conclude that humans were the infestations (viruses) on board the ship Discovery and began to take extermination actions.

The word "hacker" appeared in the 1960s to describe computer enthusiasts who began to write early versions of infectious software as a test of their ingenuity. That kind of activity led to the "cold wars" at AT&T's Bell labs and Xerox where programmers matched wits with each other by developing programs that were self- replicating and had the purpose of "consuming" programs of opposing programmers. If these high-tech "king-of-the-hill" games got out of hand, the computers were simply shut down to kill the programs in memory and then powered up again.

By the 1980s, several types of malicious programs existed. Worms could burrow their way into systems to manipulate or destroy data, however they could not replicate themselves. Likewise, non- replicating programs called Trojan Horses were destructive programs that were disguised as innocent applications. When those

applications were run, the vicious side of their behavior was invoked. Trojan Horses, called logic bombs, with programmed triggers to "go bad" when certain conditions were met also began to appear.

In the 1980s, the public became aware of the malevolent side of the computer explosion. The movie "War Games" in 1983 taught us all about the possibilities of breaking into even the most secure computer systems with devastating results. The word "virus" was linked to nasty computer programs that replicated and articles were actually written with "recipes" for virus programs. In 1986 and 1987, respectively, the Pakistani Brain and IBM Christmas viruses appeared. And, in 1988, a young Cornell student named Robert Morris created and unleashed the highly publicized Internet Virus that cost an estimated $98 million in direct and indirect costs by shutting down a national network and many large computer centers across the country.

As we entered the 1990s, virus activity on small systems and networks were a fact of life. Perhaps the single virus that caused the most public fear was the Michelangelo Virus, set to go off on the anniversary of the famous artist's March 6, 1992 birthday. It's mission was to overwrite data in the first several sectors of a hard disk, making it inaccessible (a somewhat disappointing event). I suspect it was totally responsible for the purchase of many hundreds of thousands, if not millions, of dollars of anti- virus software. Because of the widespread publicity, however, there were only relatively minor incidences of damage to systems. But, we have only seen the beginning of what will prove to be an ongoing battle between computer users and those evil little programs called viruses.

Forms of Viruses

There are fundamentally two types of computer viruses and a third that is a hybrid combination of those. The first type replicates its code and inserts the copies into other executable files. Viruses of this type infect application software files (e.g. wp.exe for WordPerfect) and operating systems (e.g. command.com). They don't care if the program belongs to a spreadsheet, database, or word processor package; the mission is simply to infect executables. When those infected programs are run, the virus will be activated and loaded into RAM. Then they can infect other programs or do some dirty work like change data in files, erase files, or flash some silly message on the screen. However, you may not notice anything wrong for some time.

The second type of virus replicates itself and infects the boot sectors of disks. This type spreads the fastest because it can be carried by floppy disks (the biggest culprit for spreading viruses) whether there are executable files on the disk or not. For this reason, these viruses multiply quickly, infecting hard and floppy disks throughout an organization. We are now seeing hybrids of these viruses that can also infect executable files, thereby increasing the speed that a virus spreads.

Phases of Viruses

All viruses exhibit behavior that can be classified into two phases: replication and activation. During the replication phase, the mission is to remain as hidden as possible while maximizing the number of infections of hosts (executable files and disks). Therefore, viruses tend not to interfere with normal operation during this phase. They could even be called careful, in the sense that they will even sacrifice some possible infections in order to go unnoticed. For example, each time the virus is activated, it might only infect a fraction of the candidate files. It knows which ones were previously infected because it embeds a signature string in the file. Or, the virus may not infect very small executables because the increase in file size will be more noticeable.

The second phase of virus behavior is called the activation phase. This is when replication may cease and trouble begins. The virus then becomes public by showing its secret purpose of silliness or destruction. This phase is triggered by some formula such as after a fixed number of infections were accomplished or a certain system date is reached (e.g. Michelangelo). Keep in mind, however, that the clones of the viruses are also out there replicating and waiting for some trigger to activate them. If no one wrote any more viruses, we'd have them around for a long time anyway.

Virus Detection

There are three types of virus protection software on the market: signature scanners, checksum scanners, and access monitors. Signature scanners check for previously identified strings of virus code, called signatures, and informs the user which one(s) they have. The same program, or another program in the package, can then vaccinate for the particular virus(es). Checksum scanners check for unknown viruses based upon the principle that viruses change the length and content of an executable file by inserting code. This type of program can be run once to check each executable file against value previously calculated and kept in a table or can be kept memory resident to check each program prior to execution. Access monitors are memory resident, constantly watching for and interrupting viral behavior, such as attempts to format the hard disk or write to an .exe or .com file. A few problems can arise from memory resident virus protection. One problem, of course, is that they take up RAM that might not be available or may cause problems in running other software. And, if you recompile programs on an ongoing basis like I do, every time you try to run a new version of a program the virus protection will yell at you.

These protection programs, perhaps all three types, should be installed on computers and run every time the computers are booted. In addition, prior to using any floppy disks, scan them for viruses. These simple steps will give a great deal of protection.

Virus Prevention

There are many things that you can do to prevent and detect early warnings of virus infestations. First, of course, you should invest in an anti-virus software package. Most current programs have all three types of protection mentioned above built in; you simply choose which ones you want to run. The minimum you should have is a signature scanner. By the way, don't forget to register the software so you can get low or no cost updates to keep current with your virus protection. These programs should be activated by your autoexec.bat file so they run prior to any other activity on the computer. Yes, it will take a little time. But, I can scan about 70 MB of files on the hard disk of my 386/33 system in about 1 minute, well worth the time. If you're on a LAN, you should periodically scan the fileserver as well. That may take several minutes, but is even more important within a multi-user environment.

Remember, floppies are the enemy when it comes to virus infestations (Schlack, 1991). Scan EVERY floppy prior to use. You might trust the providers, but you don't know where their disks have been. This includes shrink wrapped purchased software; there have been several cases where manufacturers have innocently shipped infected programs. Once scanned and determined clean, write- protect the original disks and make copies for use. Also, if you buy computer systems with preformatted hard disks, boot the systems from a previously cleaned, write-protected system floppy (you should have a few of these around with a signature scanner on them as well as the operating system) and scan the hard disk. There have been cases where these have been infected at the factory. Also, if you receive new executables by downloading them from bulletin boards or sent via e-mail from colleagues, immediately isolate those files on floppies and scan them for viruses. And, of course, back up your clean files daily for restoration in case of some catastrophe, whether or not virus related.

Finally, all users of PCs and LANs must be educated regarding the potential destruction caused by viruses and the simple steps to avoid such destruction. The best of procedures can fall apart if users continue to introduce infected disks and files to your system.

Recovery from Viruses

What do you do if a virus attacks and your system exhibits viral behavior? First of all, don't panic! Then follow some simple steps: (1) turn off your PC as soon as you suspect there's a problem, (2) boot the system from a clean, write-protected floppy with a signature scanner program on it, (3) scan the hard disk(s) to detect viruses, (4) repair any infected files and the boot sector, (5) scan the hard disk again to make sure it's clean, (6) scan and repair every floppy in sight, including backups, (7) attempt to isolate the infection source (floppies and/or other PCs), (8) reboot from the hard disk and immediately scan from a write-protected floppy for infected files on the hard disk and in RAM. If you still have a problem, do not hesitate to call in some professional help to destroy a virus or recover from its evil deeds.

Summary

Computer viruses are with us to stay and will become more of a threat as time passes. All PC professionals and users must practice safe computing by using the best and most current techniques to detect and prevent viral behavior. The alternative, is disaster for individual systems, LANs, and entire organizations whose irreplaceable data might be lost forever.

REFERENCES

Burger, R. (1991). Computer viruses and data protection. Grand Rapids, Abacus.

Clarke, A. C. (1968). 2001: A space odyssey. New York, The New American Library.

Denning, P. J. (1990). Computers under attack: Intruders, worms, and viruses. New York, Addison-Wesley.

McAfee, J. (1989). Computer viruses, worms, data diddlers, killer programs and other threats to your system. New York, St. Martin's Press.

Schlack, M. (1991, October 15). How to keep viruses off your LAN. Datamation, pp. 87-90.

Taylor, W. (1992, February). Virus-proof your pc. PC/Computing, pp. 122-137.

Developing a Lan-based Student Information System

Dr. Larry Jones, Kent State University, Geauga Campus
Dean & Asst. Professor, Computer Technology

Mel Anthony May, Kent State University, Geauga Campus
Director of Student Services
14111 Claridon-Troy Rd.
Burton, OH 44021

Background

The University

Kent State University is a large, multi-campus system covering seven counties of northeastern Ohio. The central campus is located in Kent, Ohio. It offers degrees at the baccalaureate, masters, and doctoral levels and has an enrollment of approximately 23,000 students. Kent State's Regional Campus system includes seven campuses with a total enrollment of approximately 9,000 students. The sizes of these campuses range from an enrollment of 2500 students at the Stark Campus in Canton to an enrollment of 634 students at the Geauga Campus in Burton. The regional campuses offer two-year associate degrees and baccalaureate course work for programs at the Kent Campus and transfer to other institutions.

The Geauga Campus

The Geauga Campus is the smallest two-year institution in the State of Ohio. At the associate degree level, it offers only four degrees: accounting, business, computer technology, and general studies. The enrollment dropped significantly during the early 80s to a low of 225 in 1986. There were serious discussions about closing the campus at that time. However, the enrollment has since rebounded to an all-time high of 634 for the Fall 1991 semester.

Prior to implementing local area networks (LANs), the student information system was, at best, in a shambles. Important data was not maintained at all or it was all over the place; on a VAX mini- computer, local hard drives, floppies, and mostly in unorganized paper files. We believe that networking the entire campus, both administrative and academic areas, has allowed the campus to clean up its information systems, maximize its yield rate with prospective students, target new markets within business and industry, and improved the retention rate through computer support of the entire curriculum.

Computer Systems in Place

The Kent Campus has an IBM 3090, IBM 4381, and two VAX 11/780s. There are over 30 LANs; however, they are not linked together through a common backbone. All Regional campuses' VAX systems (11/750s and 11/780s) are linked to the Kent system via JNET and DECNET communications software. Primarily, that link was established for computer technology faculty use, operating systems updates and diagnostics, and uploading local campus registration data. That linkage also provides access to BITNET. All campuses are also linked through dedicated phone lines to the Kent Campus IBM 3090 for on-line library and administrative systems lookup access.

At the time of my arrival as Dean in January, 1987, the campus had few computer resources. For instruction, there was a VAX 11/730 that was used for the Computer Technology courses, one 286, and about six PC and XT class microcomputers. Those computers were all in one large classroom that also housed six dumb terminals for the VAX. For administration, there was one terminal to the VAX in the main office used to run an old COBOL program for registration that was written at another campus. There were no microcomputers for use in administration.

Although there are a few LANs scattered throughout the Regional Campuses, by 1990 the Geauga Campus was the only one with a total campus network. There are two Novell Netware 386 3.11 LANs, one each for academic and administrative use. The two LANs are bridged together through a dedicated microcomputer to facilitate their management and provide the capability of using either network from any workstation. There are approximately 80 workstations on the LANs, all either 286s or 386s. Three academic labs, one with all 386 systems, house a total of 44 workstations. Each network has a 486 tower system for its fileserver. Both fileservers have 650 MB hard drives, one has 8 MB and the other 16 MB of RAM.

All full-time faculty, administrative, and staff members on campus have 286 or 386 workstations at their location and communicate through the use of e-mail and shared data directories. E-mail, in particular, was a catalyst that led to 100% LAN use by administrators, faculty, and staff. Regarding improved productivity in the LAN environment, we've had similar results to Oregon State University, where it was found that "because e-mail has no formal conventions, much more real communication occurs, yet memos and other paperwork have been reduced by an estimated 95 percent" (Scott, 1991, p. 54).

We have standardized on WordPerfect 5.1 for word processing, WordPerfect Office 3.0 for E-mail, Lotus 1-2-3 3.1 for spreadsheets, and dBASE III Plus for databases. We recently purchased a site license from Borland, so all of their software is also on the LAN. Everyone has access to those and a variety of other application software packages. For output, there are eight laser printers located throughout the building in labs and general office areas. On the administrative network is a complete student information system application that was designed, developed using Clipper, and implemented in-house. The VAX has been upgraded to a 11/750, but it's mostly used as a communications conduit to the rest of the University and outside world.

Strategic Need for Networked Student Information System

In the old days, approximately a decade ago, organizational information processing was hoarded by "a few individuals ordained into the computer priesthood" (Holub, 1987, p. 1). However, microcomputers have changed the way institutions design and manage their information systems infrastructures. Having experienced the utility of stand alone microcomputers, users look for ways to share software, data, and peripherals. Local area networks accomplish these objectives (Nath, 1990). Without LANs, according to Holub (1987, p. 8), "proliferation of independent computing devices with ever increasing database storage capacity, without active management supervision, could undo the benefits of the common information base."

Through the use of LANs, productivity and efficiency can dramatically improve within administrative and staff areas. Nath (1990), as a result of a study of over 200 LAN end users, reported that the quality of work life in areas of improved productivity, making jobs easier, more reliable data access, and improved efficiency all ranked high with a large majority of personnel. With the additional benefits of enhanced communications among all campus constituents, a strong case can be made for implementing campus-wide LANs. The LAN, in fact, has raised it's status from a technology toy; it's now a powerful organizational tool (O'Leary, 1990). This is particularly true at a small campus where limited resources should lead us to view information technology resources as a major strategic asset that can be utilized to maintain a competitive edge in the marketplace (Oliva & Khosrowpour, 1989).

Common strategic objectives of higher educational institutions could be placed in three categories, the three Rs: Recruitment, Retention, and Revenue generation. All areas require that institutions, particularly two-year institutions, must strive to become more dynamic, flexible, and responsive to the needs of prospective students, current students, and business and industry. To meet those defined needs, institutions need quick and easy access to institutional data and financial, physical, and human resources (Clemons, 1991; O'Leary, 1989).
Administratively, information systems developed for LANs can greatly improve the success, or "yield" rate, of turning prospective students into enrollments. According to Stephens (1989), one fundamental motivation of implementing LANs is to reduce or eliminate the amount of paper floating around and among offices so efforts can be focused on activities that maximize the return on investment of time and resources. Using centralized

databases, tracking systems, and automatic personalized inquiry response and follow-up materials, institutions can ensure that they hit the narrow window of opportunity when the prospective students' interest levels are at the peak. Likewise, those same tools can facilitate extensive tracking to minimize the loss of enrolled students due to the lack of timely problem identification or simply a lacking of adequate attention.

Planning the Student Information System

Dwight D. Eisenhower is credited with stating that "plans are nothing; planning is everything." "Without planning and a vision of what networking can achieve, the institution may never be pointed in an appropriate direction" (Madron, (1991, p. 60). The key to a successful networked student information system implementation is that process of planning throughout the information system life cycle. The planning process should involve every potential end user in exercises and discussions that assist them in thinking backwards from blue sky wants and wishes of potential output and uses of the system to the data necessary to drive it. The needs of the users and their customers, the students, should be the driving force behind information system design and implementation. Some end users have unrealistically high expectations of systems. Others will limit their thinking to doing the same old jobs in the same old ways, only with computers.

The importance of total involvement in planning is to get these disparate views out on the table for discussion. Also, the membership of any design team should include the world's greatest experts, those closest to the work. The question that needs to be asked throughout the planning stages is "how can we change what we are doing; how can [we] get more leverage gain out of these expensive paperweights" (Stephens, 1989, p. 41) to take full advantage of networked information systems? It is important to early introduce concepts of enterprise computing, which "is a matter of people talking to each other, as well as systems. Not just open architectures, but open minds too" (InfoWorld, May 6, 1991, p. 51). Motivation is the trickiest of computer questions (Emmett, 1983), but this kind of discussion and overall personnel involvement in the planning stages can be a major motivating factor as well as lead to solutions that solve the identified problems.

If a thorough job is done with planning, the information system software design, development, and implementation should be easier. However, plenty of homework should be done at this stage before investing time and money in hardware and software; its future success will depend upon decisions you make at the beginning (Strehlo, 1988).

Implementation of the network information system solution, once again, requires total involvement of the end users. That involvement must include sufficient training to facilitate the conversion from old work methods to the new automated environment. Once implemented, there needs to be an ongoing process of maintenance, evaluation, enhancement to meet newly defined problems, and planning for future growth. That effort should utilize a Total Quality Management (TQM) approach, where "all employees, every day, commit to improving the quality of their service so that customer's [students] needs are not only met but exceeded" (Coate, 1990, p. 91).

The implementation of TQM at the Geauga Campus was facilitated by pre-existing conditions. To begin with, a strong spirit of teamwork and a lack of inter-departmental barriers had developed as a result of the nature and the history of the campus. A small staff, all housed within one main office, is responsible for all the student service functions on campus. There is little specialization or clear division between staff responsibilities: everyone wears several hats. In addition, poor enrollments in the early 80's led to a strong feeling of mutual support for mutual survival, along with a customer service philosophy that emphasized the importance of each student to the survival of the campus.

The TQM process began with the campus defining its mission and market: who our customers are and what it is that they expect from us. The University has an overarching mission statement; we applied it to Geauga Campus. The application was facilitated by two processes:

1) A five year strategic plan for the campus.
2) Cooperative goal setting by the administrative staff.

Drawing upon Deming (1986), these processes established constancy of purpose, gave the campus a common language of quality, and drove out fear by clarifying how our performance would be evaluated.

TQM is a significant cultural change, not simply a management technique. It requires that leaders understand the mission of the organization and be able to communicate it to others--to take the word quality and put it under their hats. Commitment to this shared vision for the organization becomes part of every operation that is undertaken--quality is not the result of the process, it is the process. It is this emphasis on excellence that makes TQM particularly appropriate for applications in higher education. As Coate (1990) states,

> Manufacturing processes are far more predictable and controllable than the learning process. However the service areas...all have their counterparts in industry and should easily lend themselves to training in precision performance.

Reaching the same conclusion ourselves, the development of a LAN based student information system was launched.

The Geauga Campus Student Information System

To maximize the productivity of and communications among it's limited personnel, all faculty, administrators, and staff have been networked during the past two years. In addition to the normal benefits derived from network implementation, the campus defined a critical need for a network application that would allow comprehensive student tracking, eliminate existing data accuracy and redundancy problems, and provide timely access to centralized information.

Through a planning effort involving all users, a student information system was designed to meet the specific needs of the Geauga Campus. As the new system was being developed, communication flowed in both directions regarding its progress. A pattern emerged whereby as each module was written it was immediately demonstrated for the staff, who provided further input and suggestions for refinement. The refinements (wherever feasible) were incorporated and the improved module redemonstrated until the staff was satisfied with the final product. This process of constant quality improvement and innovation led to a system that far exceeds the previous registration system, meets the present information needs of the campus, and has the flexibility to the adapt to future requirements. Designing the system in modules had the advantage of allowing each to be implemented as it was completed. This made the implementation a gradual breaking-in, as opposed to an overnight change. More importantly, it allowed us to begin harnessing the front end power of the new system (recruiting and admissions) almost immediately and to begin collecting and cleaning the data that the final system would require.

Although most modules have already been developed and implemented, the project will continue to evolve over time. We call this student information system "cradle-to-grave" because it will ultimately track potential students starting with the first point of contact in our recruiting efforts and then through admission, registration, graduation, and alumni activities. Included are subsystems to support the following student related activities:

- Recruiting & Automatic Follow-up
- Admissions
- Basic Skills Testing
- Under-prepared Student Tracking

- On-line Registration
- Invoicing
- Financial Aid
- Probation Student Follow-up
- Alumni Follow-up & Fund Raising

Additional subsystems required to support these programs and meet other important needs were developed or are being developed in the following areas:

- Academic & Fee Schedule
- Course Inventory
- Personnel (including part-time pool faculty and student employees)
- Budget Analysis
- Bookstore
- Non-credit Offerings
- Business & Industry Programs
- Special Events
- Facilities Utilization

All of these subsystems are tied together through the use of bar, pull-down, and pop-up menus. The system was developed and compiled in Clipper (currently updating to version 5.01), with much of the menus, input screens, and maintenance functions generated through an application generator software package called Genifer. The use of an application generator facilitates the implementation of changes within a short time frame. A networked student information system is a very dynamic entity. The system will constantly evolve to meet users' and students' needs and react to demands of a competitive environment.

Summary

As a result of an extensive planning effort, the Geauga Campus now has a student information system that meets our current needs to track students and is flexible enough to allow future growth and development. Because of their involvement throughout the planning and development process, both administration and staff have embraced the system with a great deal of enthusiasm. And, extremely important for a small campus with limited staff, we now have ready access to accurate, non-redundant data and have become much more productive because of its availability.

References

Cles, E. K. (1991, January). Evaluation of strategic investments in information technology. Communications of the ACM, 34, 23-36.

Coate, L. (1990, December). TQM at Oregon State University. Journal for Quality and Participation, 90-101.

Deming, W. E. (1986). Out of the crisis. Cambridge: Massachusetts Institute of Technology.

Emmett, A. (1983, December). Overcoming computer resistance. Personal Computing, 80-93.
Enterprise Computing. (May 6, 1991). InfoWorld, p 51.

Holub, D. L. (1987). Guidelines for using microcomputer technology in higher education. Cleveland, Ohio: The National Center for Advancement in Education Through Technology.

Madron, T. (1991, June). Enterprise computing in higher education. T.H.E. Journal, 60-65.

Nath, R. (1990, Spring). The impact of local area networks on users and their work: A field study. Journal of Microcomputer Systems Management, 2, 15-23.

O'Leary, M. (1989, March). Home, home on the LAN. CIO, 16-26.

O'Leary, M. (1990, July). The gathering of the LANs. CIO, 46-54.

Oliva, L. M., & Khosrowpour, M. (1989, Spring). Microcomputers: Strategic tools for the 1990s. Journal of Microcomputer Systems Management, 1, 1-7.

Scott, G. (1991, June). E-mail's expanding role in education. T.H.E. Journal, 54-56.

Stephens, M. (1989, March 27). Wired: How PC networks are changing the way we work. InfoWorld, 41-42.

Strehlo, C. (1988, July). A 10 point prescription for LAN management. Personal Computing, 110-118.

Human Resource Information Systems: An ASCUE personnel profile

William J. Moressi
Academic Computing
Information Management
Winthrop College
Rock Hill, SC 29733
V010550@UNIVSCVM

The purpose of this paper is to present the results of a study describing the personnel structures of computing center departments in education. ASCUE is the focus of this study. Data collected and analyzed is from a survey distributed to current ASCUE members.

Summarized, graphic information will be provided on the number of personnel in various positions, the degree and experience required for the positions, training provided, which positions are filled by students, the "essentiality" of the positions, and supporting degree programs. As a final analysis, a profile of the person-position will be made based on the summarized data.

INTRODUCTION

In June of last year Connie Hurst and I presented a paper at the ASCUE '91 conference on "An Assessment of the Training and Development of Personnel in Small Academic Computing Centers." We were very pleased to have attended our first ASCUE conference and especially pleased at the membership response to our presentation. Several members in attendance had asked us to do an analysis of the ASCUE membership, which is not limited to academic computing centers. Connie Hurst has since received her MBA, has an excellent position with a firm in Charlotte, NC and is currently taking exams for CPA certification. (Thank all of you for your support of Connie as a graduate student. This was her first presentation before an unfamiliar group and she left the conference with a tremendous "boost" in her self confidence.) I remain at Winthrop College and have taken it upon myself to proceed with the study with the help of two of my undergraduate students.[1]

This analysis is designed to evaluate the personnel resources in small computing centers, the ASCUE membership being my sample. The Information Center (IC) personnel support structure in computing is of critical importance in our educational environment. As in most academic institutions, the information technology environment is a dynamic and continually evolving entity. Changes in the environment are "forced" in that the prime directive of computing is to meet the technology needs of the faculty and students in research and teaching and administrative computing. To meet these requirements, Information Centers must respond to the demands on its resources by implementing and supporting the ever-changing and expanding information technology platforms.

The scope of this investigation is limited by intent. From data gathered through questionnaires submitted to the ASCUE membership, I will provide summarized (graphical) information on each of several categories of information.

[1]Special thanks to Scott McLellan who did so much in developing the questionnaire in Page Maker©, and with all the processes of mail merge and data entry. A note of thanks to Pam Greene for assisting Scott in the data entry.

PROCEDURES

Questionnaires were mailed to the ASCUE membership in January of this year. To optimize our response I mailed a second set in March. Although a few questionnaires are still coming in, I stopped entry in the data base on April 1.
We have 84 responses entered into our data base from 170 questionnaires (49% response). Data was entered into dBase IV® and subsequently transformed into spreadsheets for analysis and graphing using Lotus 1-2-3® and Freelance®. WordPerfect® is used for document development.

Data analysis has been focused on several points of interest. They are to determine:

I. the number of personnel in various positions in Information Centers of small computer users in education

II. whether or not a degree and or experience is required for each of the positions

III. whether training is provided for each position

IV. whether the position can be filled by students

V. the "essentiality" of the position, and

VI. supporting degree programs.

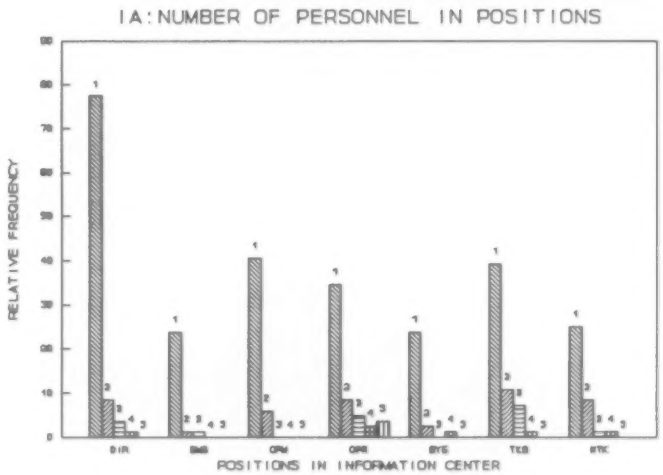
The Appendix has a brief description of each job title as it appeared on our questionnaire and as it relates directly to the following figures. Abbreviations (DIR, BMG, OPM, . . ., TCT) used in the graphs are also found in the job descriptions.

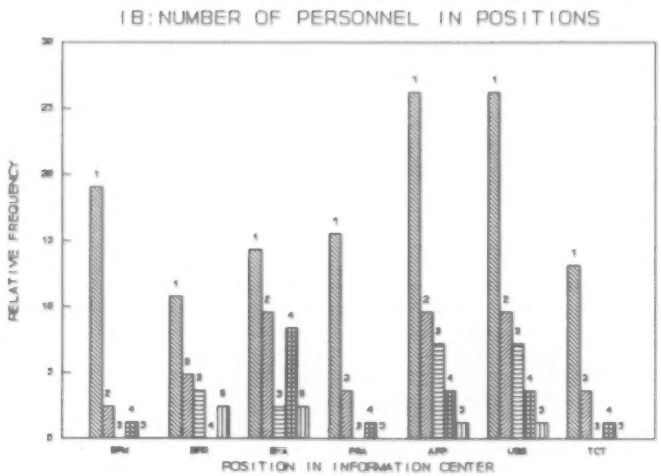
RESULTS

Results are provided using graphical distributions of the strength (relative frequency) of responses to the specific questions relating to the stated points of interest. All distributions have been rank ordered for highest affirmative response to the lowest. Darkened portions of the cells reflect the complement to one-hundred percent.

Figures IA and IB provide us with information on the number of personnel by position. Each position described, 14 provided in both figures, potentially has 5 categories or bars associated with it. The bars, from left to right, give information on the number of personnel (1 to 5) that respondents have provided. Although I obtained data for over 5 personnel in a position, the data was scattered and, in some cases, extreme. I have not included responses of more than 5 personnel in a position. Yet the relative frequency of each position does take into account all the data.

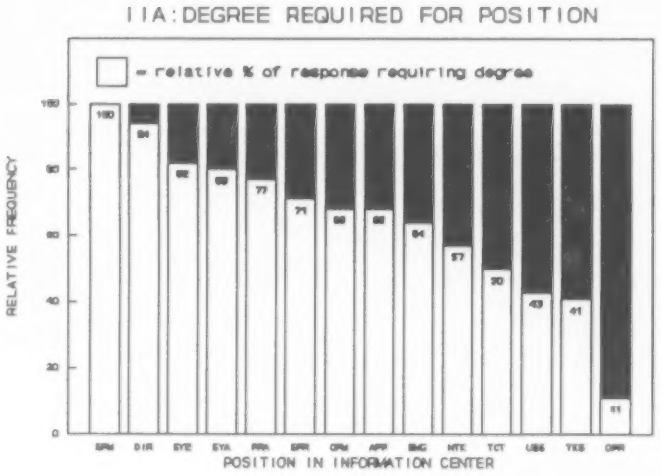
Of the 14 job positions described in Figures IA and IB, computer operators, user support and application programmers have the greatest number of personnel per position. These are also the positions respondents have indicated can be filled by students (Fig IV).

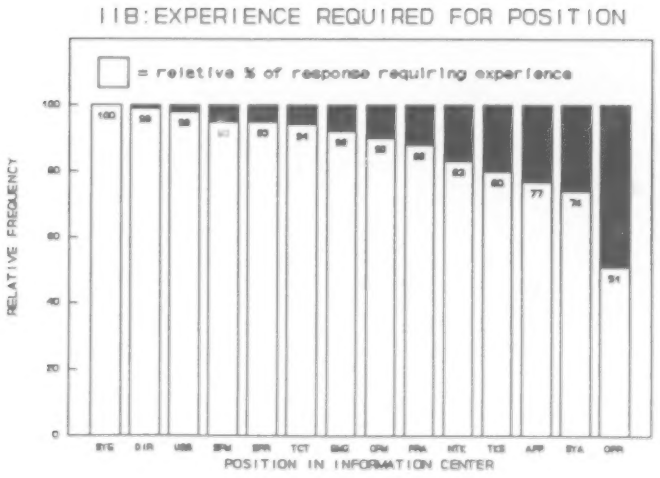




Figures IIA and IIB relate to whether a degree and or experience is required to fill the described positions.

Except for computer operators, technical support and user support, degrees are highly desirable. In specific cases, systems manager and director, a degree appears to be a strong "requirement."


IIA:DEGREE REQUIRED FOR POSITION
= relative % of response requiring degree
RELATIVE FREQUENCY
POSITION IN INFORMATION CENTER


IIB:EXPERIENCE REQUIRED FOR POSITION
= relative % of response requiring experience
RELATIVE FREQUENCY
POSITION IN INFORMATION CENTER

In all cases some experience is desired for positions in academic computing. The least experience is required for computer operators.

Figure IIIA is a relative frequency distribution for YES and NO responses regarding the question of training.

This data illustrates whether or not training is provided. The least amount of training is provided to Tele-communications Technicians and to the management positions of Business Manager, Director, and Operations Manager. Conversely, these are the positions requiring the most experience.

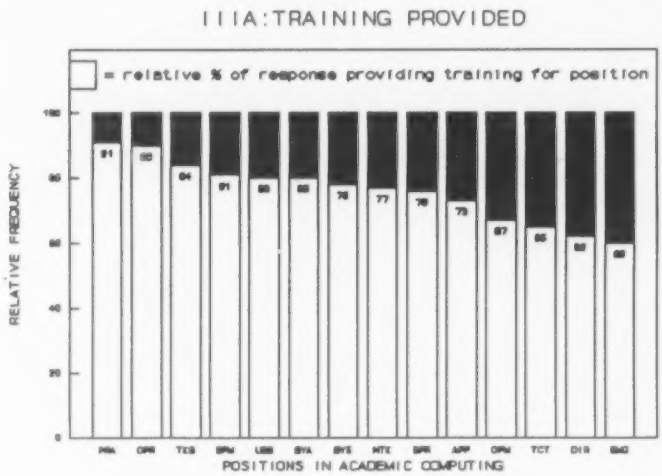
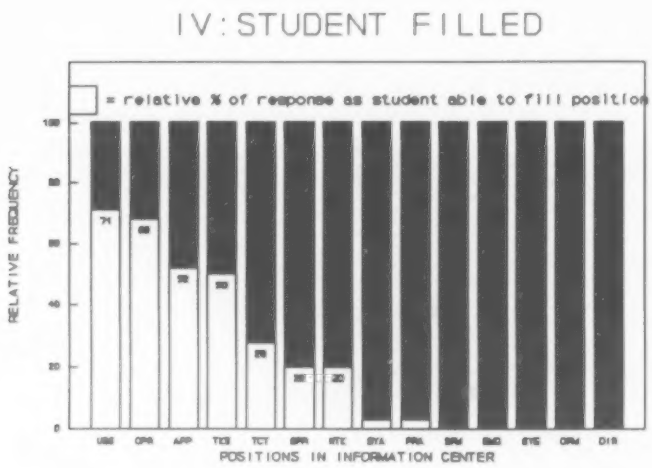

IIIA:TRAINING PROVIDED
= relative % of response providing training for position
RELATIVE FREQUENCY
POSITIONS IN ACADEMIC COMPUTING

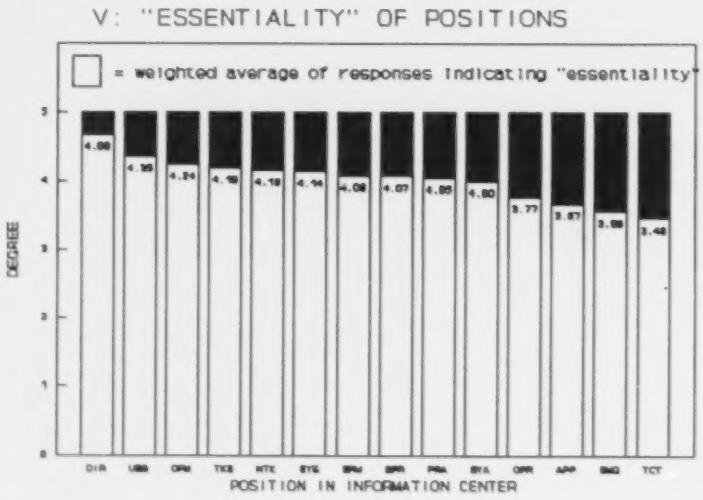
Figure IV lists the relative frequency of responses to the question as to whether the positions can adequately be filled by students.

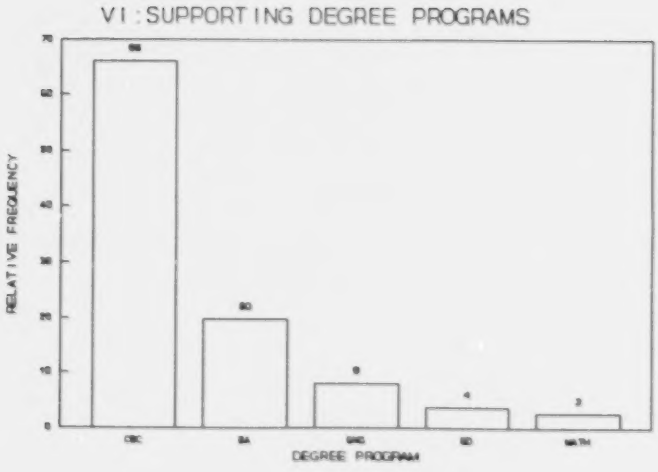

IV:STUDENT FILLED
= relative % of response as student able to fill position
RELATIVE FREQUENCY
POSITIONS IN INFORMATION CENTER

Data indicates that most positions are not appropriate to be filled by students. User Support and Operators are areas where student help is valued.

Of particular concern is the "essential" nature of a position. I define "essential" as a position that is indispensable to the functioning of the computing center for which critical services depend. Figure V provides some input on the importance placed on various positions or job functions.

The greatest degree of being essential applies to the positions of Director, User support, and Computer Operations. Note that very little indication of "least critical" was given to User Support services. These functions are listed second only to those of director.


V: "ESSENTIALITY" OF POSITIONS
= weighted average of responses indicating "essentiality"
DEGREE
POSITION IN INFORMATION CENTER



The priority of support- ing degree programs is similar to what was reported from the survey on academic computing centers. Computer Science and Business Administration programs are the most desired.

AN ASCUE PROFILE

	NUM IN POSITN	DEGREE REQUIRED	EXPER REQUIRED	TRAINING REQUIRED	STUDENT FILLED	ESSENTL POSITN
DIR	1(77%)	94%	99%	62%	0%	4.88/5.00
BMG	1(24%)	64%	92%	60%	0%	3.68/5.00
OPM	1(40%)	68%	90%	675	0%	4.24/5.00
OPR	1(36%)	11%	51%	90%	68%	3.77/5.00
SYE	1(24%)	82%	100%	78%	0%	4.14/5.00
TKS	1(39%) 2(11%)	41%	80%	84%	50%	4.19/5.00
NTK	1(25%)	57%	83%	77%	20%	4.16/5.00
SPM	1(19%)	100%	95%	81%	0%	4.08/5.00
SPR	1(11%)	71%	95%	76%	20%	4.07/5.00
SYA	1(14%) 2(10%)	80%	74%	80%	3%	4.00/5.00
PRA	1(15%)	77%	88%	91%	3%	4.06/5.00
APP	1(26%) 2(10%)	68%	77%	73%	52%	3.67/5.00
USS	1(26%) 2(10%)	43%	98%	80%	71%	4.56/5.00
TCT	1(13%)	50%	94%	65%	28%	3.48/5.00

The above table provides a summary profile of each of the fourteen positions. In column two, I have arbitrarily listed the percentages of "second positions" when they had a response of 10% or greater. The last column, that of "essential positions", I have provided a ratio of the weighted average(numerator) over the "perfect" score of five. Five is the highest response for "essentiality."

APPENDIX

BRIEF JOB DESCRIPTIONS FOR EACH POSITION AS THEY APPEAR IN THE QUESTIONNAIRE.

1. **Director(DIR):**
   Plans, organizes, actuates, evaluates, and controls activities of computing center in coordination with appropriate administrative officials.

2. **Business Manager(BMG):**
   Assists director in the management and control of the administrative operations of the center; prepares payroll and maintains personnel files.

3. **Operations Manager(OPM):**
   Directs and organizes computing center operations; coordinates flow of work and operational procedures; hires and schedules operators.

4. **Operators(OPR):**
   Keep scheduled hours for user access to computing center facilities; take supplies inventory; performs minor maintenance and back-ups; controls printers and insures output routing; generally oversees system operation.

5. **Systems Engineer(SYE)**
   Evaluates and specifies new technology, maintains the operational efficacy of all systems; provides computer systems security and contingency recovery; supervises tech support, network support, and telecommunications.

6. **Technical Support(TKS)**
   Installs computer and communication hardware; maintains and does minor maintenance on all computer and peripheral hardware.

7. **Network Technician(NTK):**
   Does minor maintenance and follows inventory control procedures for assigned network and communications equipment; reviews technical data on communications equipment upgrades; monitors communication equipment and performs diagnostics to determine problems and effect necessary resolution.

8. **System Programming Manager(SPM)**
   Oversees system development; responsible for installing and maintaining operating system revisions and upgrades; develops coding for ease of use of computing systems.

9. **System Programmer(SPR):**
   Installs and maintains the operating systems software; installs and tests applications on host systems and fileservers; monitors systems performance; provides technical interface to application, communications, and production services.

10. **System Analyst(SYA):**
    Develops computing solutions to real-world problems; provides interface between management, users, vendors and programming staff.

11. **Programmer Analyst(PRA):**
    Provides analytical and programming support; assists in supervising the programming projects; performs special tasks as assigned such as documentation of user support systems and libraries.

12. **Applications Programmer(APP):**
    Assigned to work in close coordination with programming manager to develop, document, maintain, and update application code.

13. **User Support Services(USS):**
    Assists users with software and programming questions and problems. Provides assistance to users when making purchases, installing new equipment, and using new products.

14. **Telecommunications Technician(TCT):**
    Provides routine installation and maintenance of telephone communications equipment and facilities.

BIBLIOGRAPHY

Brent, Dale and Enright, William, "Method for Planning Administrative Information Systems Development," CAUSE/EFFECT, Fall 1990, Vol. 13 #3, pp.7-14.

"Center for Computing and Information Systems, Baylor University-Organizational Listing Job Descriptions," CAUSE, June 13, 1989, pp. 6-11.

Hurst, Connie and Moressi, William J., "An assessment of the Training and Development of Personnel in Small Academic Computing Centers," Proceedings of the 1991 Association of Small Computer Users in Education National Conference, Myrtle Beach, SC.

Kettinger, William J., "The Decentralization of Academic Computing: Defining Future Roles," CAUSE/EFFECT, Fall 1990, Vol.13 #3, pp. 15-23.

Moressi, William J. and McFadden C. Brown, "Students As Academic Computer Center Personnel," CAUSE/EFFECT, July 1987, Vol. 10 #4, pp. 14-19.

Richard, Jan A., "Training and Managing Student Employees in the Computer Center," CAUSE/EFFECT, Spring 1990, Vol. 13 #1, pp.34-40+.

Warlick, Charles H., 1990 Directory of Computing Facilities in Higher Education, The University of Texas at Austin Computation Center, 1990, pp. B15-b313.

Real-time On-line Billing System For a University Environment

Balwant G. Singh
Paul G. Masucci[2]
Department of Administrative Computing
University of Maryland Eastern Shore
Princess Anne, Maryland 21853-1299

ABSTRACT

An in house developed and maintained registration billing system will be discussed. The billing system was designed to produce student bills from registration information, dynamically. The bill is comprised of charges for tuition, board, lodging, and fees. If changes are made to a students schedule or status, a new bill reflecting the changes can be immediately displayed as well as printed. All transactions related to a student bill are stored with appropriate audit trails. The system produces several audit reports, provides look-up and posting capabilities, and integrates with the student information system.

The programs comprising the billing system are written in COBOL for the HP3000. The system has been in operation at the University of Maryland Eastern Shore for the past three semesters.

INTRODUCTION

The billing process at the University of Maryland Eastern Shore (UMES) had become a real burden to operate. The system was characterized by hand written documents, massive data entry, and many reporting delays. This process could not continue considering that enrollment was increasing at a rapid pace. The old methods would not meet the demands of the growing volume of students. The Administrative Computing Department took the initiative to revamp the billing process.

What follows is a discussion of how UMES improved productivity and reliability by converting a manual system into a computerized information and billing system. The subsequent topics will be discussed: a description and drawbacks of the old billing system, an extensive discussion of the new system including objectives, and programs; and a synopsis of system results, and expansion possibilities.

THE OLD BILLING SYSTEM

System Description

To prepare for the influx of students during registration, the Admissions clerk prepared bills, one each for all the pre-registered students. It required about two weeks to handwrite approximately 1200 bills. During the two week registration period, every student required some attention with regard to billing. For the majority of preregistered students, the pre-written bills were totaled and distributed. For each new student and walk-in, a bill had to be written. Over half of the students made changes which required a new bill.

The clerk produced the bill using the student's class schedule, and several computer printed lists. The voucher contained charges for course-credits, out of state fee, housing, board, general fees, laboratory fees, and other miscellaneous charges as applicable. The student then took the bill to the cashier.

Bills were manually processed by the Student Accounts Department. The receipts were totaled and batched to be entered into the system by a data processing clerk. It might take two weeks to enter, edit, and post data to student accounts. Entry did not begin until after the registration period. One or two more weeks

[2]He is currently a Computer Specialist with the Internal Revenue Service, Silver Spring, Maryland.

198

would pass before most of the discrepancies were resolved. In the mean time all information and reports requiring student data were delayed, pushing the availability of accurate information to six weeks after registration. Students were not "officially" registered until they were registered for classes and officially paid. At UMES "officially" paid meant, charges were applied to a student's account.

Time line:

Drawbacks of the Old System

In the system described, there were long lines of students waiting to get their vouchers prepared, modified or make payments. Students were generally frustrated at the lack of responsiveness. Posting was delayed up to two to three weeks after the registration period. Consequently, there was no way of knowing the number of students who had paid and were thus "officially" registered. An audit of student records could only be done during the summer, due to lack of staff, to determine which vouchers were paid, missing or voided. Students would have to carry their copy of the bill around for over a month to be guaranteed services such as board, library.

THE NEW BILLING SYSTEM

Objectives of the New System

To generate a new system, several elements had to come together. The first step was to define the problem, and set objectives. The second, determine if the solution was technically possible. Finally, whether it was feasible in terms of expertise and resources. After painstaking discussion with all involved parties, several objectives were set.

1. Automate the manual billing system.
2. Increase productivity of the billing and student accounts systems.
3. Improve accuracy in all aspects of billing function.
4. Instant identification of paid students.
5. Renovate the current student accounts system.

In proposing the new system, several basic assumptions needed to be followed. The system was to allow students to be both registered for classes but not necessarily paid. Instead, students were allowed to be registered "officially" if charges were placed on their account. Secondly, students could not be deregistered if they had not paid in advance of the semester start or at any time unless approved.

Overview of the Current (New) Billing System

About one month before a semester ends, there is a period of pre-registration. A majority of students who intend to return for the next semester will pre-register for classes. Three to four weeks before the new semester starts, the bills are produced reflecting the previous balance and all the new charges based on the information in the Student Informatiom System (SIS). The new charges may include all appropriate fees, out of state fee, and other charges such as board and lodging. All these bills are printed at one time and mailed to the students at a designated time.

Upon receipt of the bill students can pay with the mailer provided, before classes begin and be "officially" registered. Students are encouraged to pay in advance but are not required to. If a student has financial aid or can not pay in advance, that student may bring their materials in during the registration period. The students who had their bill could go directly to the Student Account Receivable (SAR) station to make payment.

Upon payment of the bill, the cashier updates the student billing record and sets the payment flag for official registration status. After the flag is set these students are considered "unofficially" registered, implying that their billing paperwork has been collected but not posted to their account.

The student may make changes to their schedule which may result in a change in the bill amount. In such a case, the student is given an adjusted bill and the old bill is canceled. If the student has paid before making the changes, a credit memo is issued to the student, to net out the paid amount, and a new bill is created. Proper audit trails are kept for each transaction. The bills data is entered to the SAR system during registration.

System Components

The system comprises tables, databases, and several programs written in COBOLII [1]. They are executable on HP-3000 minicomputer. The billing system is menu driven and is detailed below:

A. TABLES

Several tables are required for the billing system. Tables, such as fixed charges, lab fees and message tables, are used as input by the billing subroutine. These tables must be updated in advance of the semester processing. Formal processes are in place to authorize table updates. The billing subroutine produces an output table which contains codes and charges. The charges depend on the student registration information. This table is eventually passed back to the calling program.

B. DATABASES

The bill is created from information drawn from several SIS databases: student accounting record, class schedule, residency status (in-state or out of state), boarding, and lodging. The output related to the bill is stored in two data sets in a separate database. These sets are used for reporting, audits and data tracking.

C. PROGRAMS

Billing Subroutine

The first step was to determine if charges could be derived accurately from system data. A subroutine was written to retrieve data from student databases such that charges could be generated. This program led to the construction of several tables and data sets to create an environment of reliability. This program was to be used as the engine to several other programs. The program was refined to minimize processing time as the HP3000/48 was at its peak use during registration.

There are two billing subroutines. The function of these programs is to translate a social security number and a term into a table of student account charges. These charges are then passed back to the calling program. The subroutines are used by the bill generator, bill modifier, look-up, and audit programs. Since different algorithms are needed for spring/fall and summer registration, two subroutines were written.

Bill Generator

The bill generator program is designed to produce the bills for students for a specific term. Before bills can be generated several processes must be completed.

1. All financial aid and SAR transactions for the semester must be entered.
2. Room and board information for the semester must be entered.
3. The fixed charges for the semester must be loaded.
4. The lab fee table must be up to date.
5. The semester message and due dates for bills must be loaded.
6. The honors, and gold-ID flags must be updated.

After these process requirements are satisfied, a program is run to generate a file containing social security numbers of preregistered students. Then the bill generator program is executed using this file as input. The program generates a bill for every SSN giving a unique voucher number to every bill. The program picks up the existing balance from the SAR system, and adds charges to produce the net amount to be paid by the student. A direction message, date and due date is included on the bill. The bill generator provides the option to produce bills for mailing to the students local or permanent address.

As each bill is generated, information is kept in two data sets for audit purposes. One set contains all information about the bill i.e. SSN, term, voucher number, voucher status, date, total amount, index value, and the reference number. The second set includes detailed information of each charge including SSN, term, voucher number, charge code and charge amount.

The bill is printed on a seven part form. The form comprises a top sheet, mailer, two copies of the bill and a return envelope addressed to the SAR Office of the University. The top layer has the full information printed and is retained by the Office of Admissions. The student is advised to return one copy of the bill along with payment in the return envelope. If no payment is due, the student is still required to return a copy to the SAR office to be "officially" registered.

One of the special features of the bill generator is that printing is done in the landscape mode at 8 lines per inch to accommodate all the necessary information.

Bill Modifier

The main function of this program is to generate bills, and bill corrections. The program calls the subroutine which generates a table of charges from current student information. These charges are displayed on the screen for the operator to review. If the student has not received a bill to this point, a bill will be printed and data will be stored. If a student already has a bill, both the old bill and the proposed new bill will be displayed. The bills are compared for differences. If the bills are the same no action is taken. If the bills are different, a new bill will be printed on a local printer and the new bill will be stored. The old bill will be canceled by changing its status flag. Both documents will remain stored as long as required by audit regulations.

In case the student has already paid the bill, a message is flashed on the screen indicating that the bill has been paid. In this case, two documents are produced: 1) a credit memo for the payment equivalent to the amount of paid bill and 2) the new bill. With the creation of these documents, the old bill is stored with a reference number of the document and the number of the credit memo, and its index is set to one. The credit memo record is stored with a reference number equal to the new bill document number and its index is set to two. The status of the new bill is set to active, its index number is set to three, and its reference number will be empty. All of this processing is done for audit purposes. The reference number, index number and status allow a trace of the document sequence.

The bill modifier program is run during the registration period. This program can produce a bill only after the student has been admitted and is registered for classes. If an attempt is made to create a bill before a student is registered, a message appears indicating that the student is not registered.

Look-Up Program

The look-up program is designed to give the SAR office or other designated offices the capability to look up information about student's SAR history, but not to alter it. SAR history contains charges and payments from the student's first registration to the present time. The user can also see the charge details of the most recent bill and the status of bills (paid, canceled, etc.), and credit memos for the present term. All of the above mentioned activity is initiated by entering the SSN of the student.

A special feature of this program is that function keys are specially defined to allow the users easy movement among the three screens.

Paid Flag Update Program

There are three programs which update the paid flag. The first program is very similar to the look-up program as described above. It displays the account history, most recent bill and its status, and gives the user the capability to update the flag from "not paid" to "unofficially" paid. This flag is updated when students pay their bills. The bill is accepted but will not be entered until later. The flag is used to identify those students who have been through student accounts but have not had their transactions posted. If the student has paid and the transactions have been posted, the user cannot alter the flag. The flag will remain "officially" paid.

The second program to update the paid flag is batch processing program. The third program to update the flag is the "posting" program. Each of these programs will be discussed in later sections.

Batch Entry And Edit

There are two programs in this category. The first is batch entry program and the second is batch edit program. The accountant in the SAR office collects the bills and credit memos from the cashiers, daily. Each cashier's transactions are separated, totaled, and grouped into batches. Each batch is called a "registration" batch and is identified by the prefix of its batch number. Registration batches as well as all other batches are given a six character batch number. The prefix of a "registration" batch has to have a first letter A or B followed by any second letter (from A to Z). The remaining four characters may be any alpha numeric making a total of 6 characters.

The entry clerk uses the entry program to load registration batch data. As batch number is entered the program differentiates whether it is a registration batch or not. The entry clerk then enters each student's SSN and bill number. When a SSN is entered the "active" bill is loaded on the screen from database. If payment is made towards the bill, it is entered; otherwise the default is set to zero. In any case, to load the information into a batch the "Enter" key is hit twice. All records are entered in this way. When the process is complete, totals are verified and a printout is produced.

If there is a mistake in entering the SSN, a message is displayed indicating that their is no matching bill. It also displays the SSN which matches that specific bill number. Appropriate messages are displayed for other similar discrepancies.

If the batch is in balance, it is released for posting. When a batch is released for posting, each bill which makes up the batch has the status updated to "released". This batch is then ready for posting.

Posting

After many batches are entered and released properly, the accounting office sends a list of the batch numbers to the Administrative Computing office for posting. As the batches are posted, the program updates the following flags: the student paid flag is set to "officially" paid; and the student bill status is updated from "released" to "posted". A printed report is produced showing the SSN, bill number, and details of posting. The SAR accountant uses the printed copy to reconcile the SAR accounts. The above described steps are designed to provide for checks and assure accuracy in all transactions.

Report Producing

There are several reports produced by the system. The major reports are described below:

Report	**Function**
Audit report	The program recalculates charge vouchers and compares them with the original ones. Only changes are listed with other necessary information on the Audit report.
List of voucher numbers	This report lists document numbers and other information in either voucher number order or voucher number within name order. A selection is made for either credit memos or charge vouchers.
Format program	This program produces a list of all active bills for a selected term.
Unofficially Paid Students report	The program produces a list of students who have paid their bills but have not been posted to Student Accounts. The list is used to reconcile "officially" registered students.
Zero Document Status report	The program lists students who have received their bills but have not paid. The list also is used to reconcile "officially" registered students.

SUMMARY

Results

Through time management study we were able to increase the productivity and reliability of student accounts procedure with some procedural changes, and access to the computer screens. Other improvements are noted as follows:

Bill Generator:	400% increase in advanced payments. Elimination of pre-writing bills. 100% reduction in processing errors.
Bill Modifier:	Puts responsibility of decision making in the hands of the responsible party. 100% reduction of arithmetic or billing errors.

Batch Entry/Edit:	50% increase in batch entry productivity. 25% improvement in batch editing due to remote printing functions.
Audit:	75% error reduction due to report program.

Future Expansion

The participating departments were very pleased with the performance of the new system. Suggestions were solicited to determine what would make the system better than it was. After in depth interviews with the involved departments, several improvements surfaced. What follows is a list of those suggestions:

- Change from cancellation of an entire bill to netting out specific charges.

- Automate batch entry.

- Allow on-line data manipulation.

- Print Class lists of registered students for instructors on the first day of classes.

- Set deferment flags with responsibility codes.

- Develop a fixed fees entry program.

These suggestions are currently being evaluated to determine whether they should be initiated.

CONCLUSION

The billing system has been regarded by users and management as a successful implementation of project objectives. Success can be attributed to several factors including user involvement and top management support.

Using techniques such as prototyping and user involvement, the people who were most essential to the systems success were empowered to contribute to its success. The users contributed a great deal to the finished version of the billing system.

Top management was able to effectively communicate to the users and students, that changes were needed to keep pace with increased enrollment, manpower shortage, and the national trend in using available technology. They convinced all parties concerned that the proposed system would improve efficiency and the working environment.

REFERENCES

1. COBOLII/3000, Reference Manual, Hewlett-Packard, CA, December 1986.

Computer Based Student Advising

Thomas J. Smith
Department of Mathematics and Computer Science
Manhattan College
Riverdale, NY 10471

This paper will describe a computer based student advising system recently implemented in the School of Arts and Sciences at Manhattan College. The methodology required to set up the system, the benefits derived from its implementation, and the difficulties encountered will be discussed.

Introduction

Manhattan College is a private, independent, primarily undergraduate institution located in the Riverdale section of New York City. The full-time undergraduate enrollment is approximately 2750, divided among four schools, Arts and Sciences (950), Business (750), Engineering (750), and Education (300).

Each of the four schools handles its own student advising. Prior to 1991, two of the four schools, Business and Engineering, had developed their own totally independent computer advising systems. The one in Business utilized the main administrative software package purchased from Information Associates (IA), while the one in Engineering was a microcomputer based system using FoxPro, a database programming package. The School of Arts and Sciences did all of its advising manually, relying on information provided by the Registrar's Office and the Assistant Dean of Arts and Sciences who kept a tally sheet on the progress of each student in the school. Each student in the school has a faculty advisor and each advisor has a folder containing relevant information for each advisee. Complicating the matter still further are several factors: the large number of majors and programs (26) available in the school, the existence of a fairly large core curriculum begun in 1985, the large number of students who enter as undecided majors, and an increasing number of transfer students. Advisors were finding it increasingly difficult to provide accurate advice on appropriate required and elective courses for various programs.

In the early summer of 1991, the Provost of the College and the Dean of the School of Arts and Sciences agreed that a computerized student advising system was an immediate priority. The author, a Professor of Mathematics and Computer Science and Director of the Core Curriculum, was asked to put such a system into place.

The first decision which had to be made concerned the type of advising system to be used. The two existing systems were evaluated. The microcomputer based system used in Engineering had several advantages; it is fast, flexible, and easy to use. Access to the system is provided by a local area network of microcomputers located in the Dean's office, the four department offices, and the offices of several faculty advisors . These advantages are offset by the need to write customized software, the expense of establishing a microcomputer network within Arts and Sciences, and the expense of purchasing software. The customized software for Engineering had been written over a one year period by one of the faculty members in the school and his son. Some of that software could have been used in Arts and Sciences but the degree requirements and programs are so different in the two schools that much of the software would have had to be written from scratch. A further difficulty with this approach is that the information in the database must be downloaded from the main computer system whenever timely information is needed. The data in the microcomputer database is only as timely as the last time it was downloaded.

The School of Business approach was to use the IA system, working with the On Course module which comes with the system. The Business approach had been developed over a one year period by the person who was then the Assistant Dean of that school. The advantages here were that no additional hardware or software

needed to be purchased since the information was accessible on-line to anyone with a terminal connected to the administrative system. Further, expenses were limited to the cost of learning the system and entering the required data. The disadvantages are that the IA system is cumbersome to use, not well documented, and not overly flexible.

After reviewing the two systems and conferring with the Provost and the Dean, a decision was reached to implement the student advising system using the IA software. The principal reason for the choice was economic; this was the less expensive way to go.

Implementation

Starting in the late summer of 1991, the author began the process of implementing the student advising system. This process can be thought of as involving two steps. The first is to enter into the computer the requirements for each of the majors in the school; the second is the production of student audits which match the requirements with the work the student has already completed. The entry of requirements is essentially the translation of the requirements found in the College Catalogue into a form which the computer can use. It is important to note that Catalogue requirements are not static. The requirements for a certain program in the 1990-91 Catalogue may differ from the requirements in earlier or later Catalogues. Thus, the entry of requirements may have to be updated on an annual basis. The Catalogue for 1990-91, which contains the requirements for the present sophomores, was chosen as the first to be entered.

The IA system imposes certain guidelines in terms of structure. For the programs in Arts and Sciences this meant that the requirements could be divided into five general categories:

(1) general degree requirements, such as the number of credits needed for a degree, the number of hours which must be completed in residence, and the minimum grade point average (GPA);

(2) the core requirements; these include all of the courses which must be completed by all of the students in the school (actually, there is a slight difference in core courses depending on whether a student is pursuing a B.A. or a B.S.);

(3) the major requirements; these in turn involve specified courses and major electives;

(4) cognate requirements, if applicable; for example, students pursuing a B.S. in mathematics must complete two years of science courses;

(5) free electives.

Each of the sets of requirements is entered via a sequence of screens within the On Course module. Once the logic of the system is mastered, the process is relatively straightforward, although the screen entry process is very tedious. Each major involves the entry of data into a large number of screens; a typical major will require more than 50 screens (or pages to be entered). The process is slowed by the lack of adequate wordprocessing capability in the system. On the bright side, some of the requirements, such as the core, which are common to multiple programs, may be copied from one program to another so that they do not require multiple entries.

The degree of difficulty for the various majors is a function of the flexibility in the requirements. Majors which allow multiple tracks are more difficult to describe accurately to the computer. Majors with a simple set of specifications (such as five required courses and five major electives) are quite easy. Once a major is described for a given Catalogue (or academic year) it is relatively simple to update it for subsequent years since most of the requirements will presumably stay the same. The initial entry, therefore, is the most complicated part of the process.

After each major was entered the requirements were distributed to the appropriate department chair for checking. This is extremely important since, in several instances, what was described in the Catalogue, was not in fact what was being enforced. This step also identified several inconsistencies in certain major requirements.

By the end of the first semester of the academic year 1991-92, almost all of the 26 possible majors had been entered for the years 1989-90, 1990-91, and 1991-92. These then describe the requirements for the present freshmen, sophomores, and juniors. No attempt was made to enter the requirements for the present senior class since the core requirements which applied to them had changed considerably for later classes.

Use of the System

The first full use of the system took place this semester, the spring of 1992. All students at Manhattan preregister in the spring for the following fall. Prior to the preregistration period, copies of all student audits were distributed to faculty advisors. A meeting was held for these advisors to describe exactly what information the audits contained and how they might be used in helping students to prepare their schedules for the next semester. These audits are generated by the computer which matches the student's major and Catalogue year with the requirements for that major and year.

The actual audit produced consists of three types of information. The first is summary information on the number of credits completed both at Manhattan and, if applicable, at any transfer institutions, the number of credits presently being attempted, and the overall GPA. The second piece of the audit lists all of the degree requirements which have yet to be completed. These are broken down by the categories (core, major, free electives, etc.) mentioned above. This section is particularly useful for the advisor. The last section of the audit is a transcript of the work completed by the student. This transcript is also organized by category and a GPA for each category is computed; the advisor now has available, in a convenient form, the student's work within the major.

An example of a typical student audit is provided in the Appendix.

The present intent is to provide advisors with these audits each semester prior to the preregistration period. The audits are timely in the sense that the information is accurate as of the date the audit is run. This will include information on any withdrawals from courses up to that date.

Benefits of the System

Faculty advisors now have timely and accurate information on student records when meeting with their advisees. This is a major improvement over the previous manual system. Initial faculty reaction has been very favorable.

The review of major requirements by department chairs done early in the process helped the chairs to spot redundancies or ambiguities in the description of their majors.

The review of Catalogue requirements resulted in the detection of a number of mistakes or inconsistencies within the descriptions provided by the school.

The production of student audits helped to catch a number of errors in student records. The most common of these were the incorrect entry, by the Registrar's office or the Dean's office, of transfer credit. Credit given for electives should have been major credit or vice versa.

Problems with the System

Certain programs, such as International Studies, with a great multiplicity of options, are difficult to describe to the computer.

Transfer credit is sometimes inaccurate, particularly if the entries in student records are inconsistent.

Working with the IA system is tedious and made more onerous by the lack of wordprocessing capability. (In entering requirements, for example, most mistakes can only be corrected by retyping the entire description; there is no provision for inserting text.)

Dealing with IA is complicated by the fact that documentation for parts of the On Course module is inadequate and all phone calls are routed to an answering system. A representative of IA will call back but, if you are out of the office at that time, then the game of telephone tag begins. Once connected to a human, however, the help provided by the people at IA is excellent. Their representatives are knowledgeable and more than willing to help. A further complication, however, is that, as of the present time, very few schools have implemented a student advising system using the On Course module. This means that certain problems which arise may be the first time that the IA staff has encountered them.

Work to be Done

Over the coming summer work will be completed on the inclusion of double majors and minors which are not yet part of the audit. Those few programs not in the system will be added. Any updates needed for the next Catalogue (1992-93) will be made.

The system as presently used is essentially hard copy oriented. Printed copies of all student audits are produced by the Computer Center and delivered to the advisors each semester. These audits can be done on-line provided that an advisor has a terminal connected to the administrative system. Not all advisors or even all departments are currently in that position. As more connections become available, faculty and department chairs will be trained in accessing the system. A long term goal will be to provide each faculty advisor with that capability.

Conclusion

Although the implementation process was somewhat tedious and time consuming, the new computer based advising system is a vast improvement over the old system. The advantages of this system will be magnified over the next few years as the School of Arts and Sciences faces the prospect of an increasing number of transfer students, compounded by large numbers of incoming students who enter as undecided majors or switch majors during their undergraduate years.

APPENDIX

Smith, Derek K 083-60-7776

Analysis of Academic Progress
SCHOOL OF ARTS & SCIENCES
BS IN MATHEMATICS
Catalog 91F, Academic Year 1991 FA

This evaluation is provided for advisement; it is not an official record.
Please report any additions or corrections to your advisor.

- - - - - - - - - Summary of Credit towards Graduation - - - - - - - - - -

REF	INSTITUTION	PASSED	IN/PROG	N/A	GPA VARIANCE
1	MANHATTAN COLLEGE		15.00 31.00		4.000
2	A.P. Program, The College Board	14.00		3.00	
		29.00	31.00	3.00	4.000

SECTION 1 - Academic Requirements Remaining
****

This analysis assumes successful completion of all course work currently in progress.

- - - - - - - - - - HOURS/GPA REQUIREMENT (MATH BS) - - - - - - - - - - -
65 hours must be completed with a minimum GPA of 2.000. The GPA's in this audit are for graduation only and not for honors.

- - - - - - - - - - - - - RESIDENCY REQUIREMENT - - - - - - - - - - - - - -
4 hours must be completed in residence.

- - - - - - - - - - - - CORE FOR SCIENCE MAJORS - - - - - - - - - - - - -
The following CORE requirements must be satisfied:

ROOTS:

ROOTS of the MODERN AGE: complete 3 courses: LLRN 203, 205, 206

RELIGIOUS STUDIES:

Choose 1 course from: RELS 300-399

Choose 1 course from: RELS 400-499

- - - - - - - - - - - MATHEMATICS MAJOR REQUIREMENTS - - - - - - - - - - -

A minimum grade of 'C' is necessary in each of the required courses (43 credits).

A. COMMON REQUIREMENTS: complete 7 courses: MATH 203, 215, 315, 407, 411, 413; CMPT 339

- - - - - - - - - - - - - SCIENCE REQUIREMENTS - - - - - - - - - - - - - -

One year (8 credits with lab) of the same natural science is required.

- - - - - - - - - - - - FREE ELECTIVES (MATH BS) - - - - - - - - - - - - -

21 hours in free electives must be completed:

Free electives must be chosen from: ASTR 090-499; BIOL 090-599; CHEM 090-599; CLAS 090-499; COMM 090-499; CMPT 090-499; ECON 090-499; EDUC 090-499; ENGL 090-499; ART 090-499; FREN 090-499; GERM 090-499; GOVT 090-499; HIST 090-499; INTL 090-499; ITAL 090-499; LAAS 090-499; MFL 090-499; MATH 090-499; PEAC 090-499; PHIL 090-499; PHED 090-499; PHYS 090-499; PSYC 090-499; RELS 090-499; SCI 090-499; SPAN 090-499; SOC 090-499; SPCH 090-499; URBN 090-499; ACCT 203-204; FIN 305-306; CIS 206; HLTH 090-499; ENGS 101; MKTG 301; MGMT 301

SECTION 2 - Academic Requirements Completed or in Progress

****

- - - - - - - - - - - CORE FOR SCIENCE MAJORS - - - - - - - - - - - - -

ENGL 110	COLLEGE WRITING	91F	3.00	A
FREN 201	INTERMED FRENCH I	91F	3.00	A
FREN 202	INTERMED FRENCH II	92S	3.00	Registered
LLRN 102	ORIGINS WESTERN CULTU	91F	3.00	A
LLRN 120	ROOTS SOC SCI - ECONO	92S	3.00	Registered
LLRN 121	ROOTS SOCIAL SCI: GOV	92F	3.00	Registered
LLRN 204	ROOTS MODERN AGE: LIT	92F	3.00	Registered
RELS 110	NATURE+EXPER OF RELIG	92S	3.00	Registered

Total 24.00 GPA 4.000

- - - - - - - - - - - MATHEMATICS MAJOR REQUIREMENTS - - - - - - - - - - -

CMPT 101	INTRO TO CMPTR SCI I	91F	3.00	A
MATH 109	HONORS CALCULUS I	91F	3.00	A
MATH 110	HONORS CALCULUS II	92S	3.00	Registered
MATH 209	HONORS CALCULUS III	92F	3.00	Registered
MATH 213	FOUNDTN OF HIGHER MAT	92F	3.00	Registered

Total 15.00 GPA 4.000

210

---------- SCIENCE REQUIREMENTS ----------

CHEM 101	GENERAL CHEMISTRY I	92F	4.00	Registered	
PHYS 101	PHYSICS I	91F	4.00	Transferred PHYS101	from #2
PHYS 102	PHYSICS II	91F	4.00	Transferred PHYS102	from #2
	Total		12.00		

---------- FREE ELECTIVES (MATH BS) ----------

CMPT 102	INTRO TO CMPTR SCI II	92S	3.00	Registered	
GOVT 203	AMERICAN POLITICAL SY	91F	3.00	Transferred GOVT203	from #2
	Total		6.00		

Reengineering Campus Administrative Information Systems

Glen W. Turney
Administrative Computing
Kenyon College
Gambier College
Gambier, OH 43022-9624

Reengineering has recently become a common term used when discussing the status and direction of campus administrative information systems. Most current administrative application software is designed around hardware and software capabilities 10 to 15 years old.

During the last 10 to 15 years, advances in computer hardware and communications networks have far outpaced advances in administrative software. Application developers are therefore in a position of playing "catchup", to build administrative computing systems, using the latest technical tools, that better serve the campus community.

At first glance, reengineering administrative systems seems to be a large task with high risks. Every administrator remembers a past computer system conversion and shudders at the thought of being faced with that experience again. But past problems can be avoided with proper communication, planning, organization, and use of new computing tools. There is an old saying that you "eat an elephant a bite at a time". The secret is to make the project "bites" as small as possible and "eat" them in the right sequence.

The discussion of the reengineering process must start with a review of administrative computing objectives and an understanding of the current status of applications systems. A review of new technology follows to provide the basis for the establishment of a new systems approach. Next, the issues of systems implementation must also be understood and, last but not least, the reengineering of administrative systems must provide a benefit to the campus community.

ADMINISTRATIVE COMPUTING OBJECTIVES

The historical objective of administrative systems has been to:

Improve the administrative efficiency of the departments of the enterprise.

This objective has been oriented to administrative departments for two reasons. First, successful computer systems are built to support and improve the operation of a specific function or organization. Since departments are the primary organization unit of the campus, current systems were built to support specific departments. The second reason is that old technology did not effectively support anything beyond a departmental unit.

But the reengineering of administrative systems should center around a much broader campus wide information objective.

Improve the administrative efficiency of the total enterprise.

Improving the flow and processing of information between departments, as well as within departments, will result in efficiency gains for the total enterprise. Current network technologies make this approach reasonable and viable.

Campus Wide Information Systems (CWIS) are currently appearing on many college campuses. A CWIS stores and displays information. It consists of news, bulletin boards, information announcements about campus activities, and other topics of general interest to the campus.

Using current network and data base management technology, a campus wide administrative information system (CWAIS) can be implemented to better support the administrative functions of the enterprise.

CURRENT SYSTEMS

But before discussing the CWAIS approach, however, it is necessary to review and understand the current status of administrative systems. As previously stated, information files and systems do support, and thus mirror, the structure and functions of the campus. Since the campus is organized and structured by departments, current administrative systems are department oriented. A brief review of current systems illustrates the need to consider reengineering.

The current departmental administrative systems are:

Department	System
Admissions	- Admissions - Financial Aid
Registrar	- Registration - Student Records - Degree audit
Student Affairs	- Student Housing - Student Advising
College Relations	- Alumni Development
Finance & Administration	- Accounts Receivable/Student Billing - Personnel/Payroll - Purchasing/Payables - Fixed Assets - Fiscal reporting (General Ledger, Budgeting, and Reporting)
Maintenance	- Shop Order - Inventory

Since current systems are department oriented, information must currently flow between departments via the paper network. The paper network consists of stacks of documents and/or reports. The documents are processed and/or generated in one department, received in another department, batched, and entered into the department system. Next, a series of batch runs are processed to update files and print reports.

It is interesting to note that the reports, generated by each department system, add to document flow within and between departments. A report from one department system may become input to another department system.

The current systems, used for department processing, are not easy to change. They were either bought by departments or built by a central group of local computer system specialists. The systems therefore must be revised by the vendor or the local computing specialist staff.

Also, the development tools used to build current administrative systems had limited functionality. When improvements in data base management systems (DBMS) and application development tools occur, they are slow to be utilized in all systems, because of the time requirements of upgrading and conversion.

In reviewing the software offerings of administrative software vendors, most offer a DBMS that is proprietary to the specific applications offered by the vendor. This means that the vendor built the DBMS and is thus required to upgrade and maintain the DBMS in addition to the applications. Therefore, additions and upgrades to the proprietary DBMS functionality are limited.

There are a number of conclusions that can be drawn from a review of current administrative computing application systems:

1. Clerical time is required, in each department, to create, receive and maintain the paper network that feeds departmental systems.

2. Access to current information is difficult because current information consists of information on the system plus the stacks of documents in transit.

3. Revisions from a vendor or central applications group are time consuming, creating an applications backlog and a continuing wider gap between systems requirements and systems performance.

4. Therefore, "shadow" systems and files appear on less expensive department PC's to satisfy information requirements not met by the departmental system.

5. This, in turn, creates duplicate information files in different departments on different hardware/software platforms.

6. Additional systems are required to maintain duplicate files.

7. Thus, additional management and clerical time is required to maintain duplicate files and systems.

8. Finally, the creation of duplicate files and systems also creates errors and file discrepancies that must be continually resolved.

Reengineering current administrative systems into a CWAIS environment provides the opportunity to improve administrative efficiency of the campus by reducing paper flow, duplicate systems, and duplicate files.

THE CAMPUS WIDE ADMINISTRATIVE INFORMATION SYSTEMS APPROACH

The CWAIS approach is different from current systems approaches. The CWAIS focuses applications development on campus wide functions as opposed to individual department functions. In the process, departmental information needs are satisfied, in addition to the achievement of campus wide efficiencies.

The first step in implementing a CWAIS is to identify file categories that mirror and support campus wide administrative functions, focusing on the customers and resources of the institution. A simple list of file categories that represents, at a very broad level, the administrative functions of the institution is:

Customers:

Prospective Students
Students

Resources:

Alumni and Friends
Employees
Facilities
Vendors
Budget Control (Finance)

The next step is to further define the file categories into file subsets. Attachment "A" contains a list of file categories and the corresponding file subsets represent the next level of definition.
You will note that these file subsets may continue to be defined into other file subsets until a manageable level of file definition is reached.

After the file subsets are defined, simple file tables of rows and columns containing fields of information are defined for each file subset. Each data field is stored only once in the system. Security levels are defined to specify personnel having update and/or inquiry capability for each data field.

Finally, processing modules are written to update and access specific groups of data fields. Processing procedures can be defined and executed, based on entry of data, thus eliminating the need for much of the scheduled batch processing. This feature is important. It provides the ability to immediately update all defined accumulation fields when the data field is updated. Persons accessing the information on-line are therefore looking at the latest accumulated information. Processing procedures may also be executed through scheduled batch processing where required.

Data fields may be updated and/or accessed by campus personnel from any terminal on the network according to security definitions. Personnel will update and/or access data fields according to their respective responsibility in the college organization. User friendly interfaces, such as windows, are available to permit users to make choices by pointing and clicking as opposed to entering codes.

It is important to note that the development of a CWAIS is an evolutionary process. Data subsets and corresponding processes may be implemented individually or in small groups. But it is important that they be implemented within the total CWAIS framework.

AN EXAMPLE - EMPLOYEE INFORMATION SYSTEMS AT KENYON

The CWAIS approach has already been successfully implemented on three file subsets of employee information at Kenyon College. The employee information file subsets are personnel, campus, and faculty information. The following outline details the problem, solution and results of the project.

Employee Information at Kenyon - The Problem:

1. Personnel duties are shared by a number of departments based on the type of employee (faculty, administration, support staff, or maintenance).

2. Personnel information additions and changes were entered on paper forms by different departments and routed through other departments via the paper network to ICS to update the computer employee file.

3. The paper flow system was not reliable causing the computer file to be incorrect.

4. Because all college departments did not trust the computer employee file, many maintained their own department "shadow" file of college employees, based on information received from various sources.

5. Each year it was necessary to send out mailings to all employees to correct the computer employee file, which is a primary input to the campus directory.

6. All faculty information was maintained on paper files in the Provost office.

7. Access to information about particular employees by management personnel was impossible.

Employee Information at Kenyon - The Solution:

1. The overall solution was to have one computer employee file that would be the official accurate list of campus employee information. This file would be updated and accessed from any department on campus, according to defined security.

2. Computer files were established for personal, work, and faculty information, with corresponding on-line update screens.

3. Validation files were set up for specific information fields to insure consistency and accuracy of data.

4. A specific group of persons were identified, according to their campus function as opposed to department, as the EMPMAS update team.

5. The EMPMAS update team members were given the ability to directly update the computer employee information files, based on the employee information responsibility assigned to their office.

6. The EMPMAS team input assignments are:

Office	Employee Type
President	Administrative Exempt
Provost	Faculty
Staff Relations	Salaried and Hourly Staff
Maintenance	Maintenance

Office	Information Type
Payroll	Address changes
ICS Operations	Email Addresses
Registrar	Faculty Voice/Vote
	Procession

Office	Faculty Information
Provost	Status, Assignment
	Review Dates
	Committees
Academic Dean	Office Assignment
Public Relations	Degrees, Associations

7. Batch processes were written to examine the file each day for changes and automatically communicate notification of changes to each member of the EMPMAS team via EMAIL.

8. Inquiry screens were written to permit general access to the file on selected information.

9. Batch processes were written to daily transfer address information from the employee file to the payroll file.

10. A daily batch was implemented to automatically create EMAIL distribution lists for all employees, selected employee groups, divisions and departments.

Employee Information at Kenyon - The Result:

1. Almost all paper flow has been eliminated from this process.

2. The EMPMAS team members have accepted a campus responsibility and have done an excellent job of accurately maintaining their information assignments.

3. Clerical time has been saved in many departments because the improved file accuracy has eliminated the need for individual departments to maintain their own files.

4. Clerical time has been saved in many departments in maintaining departmental EMAIL distribution lists.

By using the CWAIS approach, a simple solution was implemented for a complex situation. This simple solution would not have been possible without recent changes in and availability of technology.

This implementation of segments of the employee information system at Kenyon clearly demonstrates that the campus wide administrative information system approach is workable and improves administrative efficiency.

NEW TECHNOLOGY

A number of technical advances have made the building of a CWAIS possible. These advances include the campus electronic network, improved data base management technology and application development tools, faster and less expensive computer processors, and faster and less expensive data storage.

The campus electronic network is one of the most dramatic technical advances on the campus. It has literally changed the communication culture of the campus. It provides electronic communications between campus personnel as an alternative to the current paper and voice communication links.

Through the campus electronic network all persons on campus can easily and quickly communicate with each other, as well as with peers from other educational institutions. All administrative personnel on campus can also access administrative computer systems files according to defined security.

The campus electronic network is bridging the current barriers between departments thus providing the opportunity for the campus to function as one integrated department. This satisfies a basic requirement for a successful CWAIS.

In addition to the electronic network, relational data base management system (DBMS) technology is another important technology tool. DBMS packages currently on the market provide a number of important features that are essential to building a CWAIS.

1. Tools to **create and manage relational data tables** provide the ability to store and retrieve data fields in a row and column format.

2. **Data view definitions** of fields from multiple tables provide the ability to access and control data in a logical format that is different from the physical table format. Thus data can be accessed in different logical groupings but physically stored only once.

3. **Security** at the data view level provides the ability to define security for any person to access any logical combination of data fields.

4. A **central dictionary of data fields** is essential for the management of a common DBMS for the college.

5. **On-line transaction processing** provides the ability to immediately update the DBMS from any point on the network.

6. **Field level lockout** for on-line processing insures the integrity of data in an on-line environment where multiple transactions occur simultaneously.

7. **On-line journaling, backup and recovery** provides data recovery and integrity in case of a hardware or software failure.

8. **Processing procedures initiated by entry of data** into a table permits the automatic update of other data fields as a result of on-line entry thus eliminating the need for batch processing.

9. **Data inquiry** provides access to all CWAIS data for which a person has security clearance.

10. **Application development tools** consisting of:

    a. **Menu and Screen generators** to build on-line processes.

    b. A **forms generator** to print special forms and custom letters from data files.

    c. A **report writer** to generate and print reports.

Current DBMS tools are also portable from the PC hardware platform to the midrange and mainframe processor platforms. This provides the opportunity to run the same application systems on a desk top or central computer. The flexibility of selecting smaller and less expensive computers offers an additional cost savings opportunity.

Current DBMS tools also support "client-server" distributed processing. Distributed processing provides the opportunity for savings by downsizing to less expensive computers. In this environment, the data base and related processing code resides on a server. Users can access the server via the electronic network, execute computer processes and access the data base using their desktop station as the computing engine.

SYSTEM REENGINEERING ISSUES

The first, and most important reengineering issue is the support of administrative personnel for the process. It was earlier stated that successful computer systems are built to support the operation of a specific function or organization. Therefore, senior staff backing of the system reengineering project is essential because the senior staff is the one organization that represents the total campus.

In addition, the participation and support of all administrative supervisory and clerical personnel is a must. The environment must be established for all personnel to work together on teams to objectively review current practices and suggest improvements. They must be permitted and willing to think on a campus level, in addition to their department level responsibilities. After all, administrative personnel, who do the work on a daily basis, are usually the best source of ideas for efficiency improvement.

The second critical issue centers around implementation approach. The approach should be "evolutionary" as opposed to "revolutionary". There should be no "turn key" conversions. Such projects cause high stress and are not necessary given the current stability of administrative systems.

The most desirable approach is to shift to the new environment through a series of projects over the next 3 to 5 years. Each project should be as small a possible. Highly visible projects should be done first to gain the support and enthusiasm of the administrative staff. Early projects should also be selected that have a high probability of success. Finally, high payoff projects should be done as early as possible.

The "make" or "buy" decision is a third issue. The possibility of buying the software should be evaluated for each project. It is sometimes less costly to buy a wheel rather than reinvent the wheel. But it must be remembered that any system considered for purchase must serve the college information needs. The college should not change its requirements to fit the proposed system. The "buy" decision should also include consideration for outsourcing certain applications

A fourth issue is the Data Base Management Platform selection. The DBMS tool selected must have the features outlined in the above technology discussion. The DBMS tool should be selected from a company that is primarily in the business of building and selling DBMS tools. A DBMS built only for a particular application software offering should be avoided. Companies primarily in the application development business do not usually expend the funds necessary to continually advance the capabilities of the DBMS tool.

The selection of a different DBMS tool will, in all probability, be required to implement the CWAIS. It is important to note that the skills of current administrative personnel in using current DBMS tools are transferrable to the new DBMS tool. You don't need to learn how to drive again if you get out of a Ford and get into a Buick. Your driving skills are transferrable. The same is true when you use a new DBMS.

The final issue is funding. An investment will be required initially to achieve benefits in the future. A new DBMS tool, purchased software, possibly hardware, and systems development personnel will cost money. A plan must be developed that will attract funding.

PAYOFF

As previously stated the campus wide administrative information systems objective is to:

Improve the administrative efficiency
of the total enterprise.

The on-line availability of accurate information will improve the overall effectiveness, efficiency, and work tempo of all levels of administrative personnel.

The quality of work will also be improved by moving away from the current assembly line process of handling information. It has been documented by quality studies that most people, in a batch work flow environment, spend 30% of their time correcting output received from others.

This improvement in efficiency, effectiveness and quality will, in turn, result in better service to the parents and students, the customers of the college. This high level of customer service can be provided at the lowest level of cost and thus improve the competitive position of the college.

Savings can be specifically achieved by eliminating time required to:

1. Create and send documents between departments.
2. Receive and process documents.
3. Input documents into multiple clerical or automated systems.
4. Process information through multiple systems.
5. Reconcile differences in multiple files on multiple systems.

SUMMARY

Current administrative systems are primarily department oriented. The systems are stable and functioning without serious problems but contain numerous inefficiencies. This discussion has shown that there are opportunities for efficiency improvements by utilizing the CWAIS approach in conjunction with new and improved technology tools.

Attachment - A

CWAIS File Subsets

File Category	File Subset
Prospective Students	Personal
	High School
	Performance
	Activities
	Interests
	Campus Contacts & Visits
	Application
	Acceptance
	Financial Aid Profile
	References
	Athletic Profile
Students	Personal
	Housing
	Advising
	Course Scheduling
	Registration
	Campus Activities
	Performance
	Degree Audit
	Health and Counseling
	Accounts Receivable
	Financial Aid
	Student Loans
	Public Affairs
	Career Counseling
	Payroll
	Library
	Security
Alumni and Friends	Personal
	Family Profile
	Work Profile
	Activities
	Giving History
Employees	Personal
	Campus
	Faculty
	Payroll
	Public Affairs
	Salary Administration
	Accounts Receivable

Facilities	Status
	Capacity
	Technical Equipment
	Communications
	Asset Value
	Shop Order
	Inventory
	Security
Vendors	Vendor Information
	Purchase Requisition
	Purchase Order
	Accounts Payable
	Vendor Performance
Budget Control (Finance)	General Ledger
	Budget
	Financial Reporting
	Projects and Grants

Earning "Monopoly Rents" Through Innovative Programs Using a PC

Mary Wall
Atlantic Community College
5100 Black Horse Pike
Mays Landing, NJ 08330-2699

ABSTRACT:

When an organization leverages critical resources the result can be sustainable economic advantage, what economists call "Monopoly Rates." We at Atlantic Community College have used our key compentencies--our academic standards and our strong relationship with the community, to provide innovative programs using the personal computer. Some programs are:

(1) a six-month computer programmer series which, while non-credit, articulates to college credit at completion.

(2) credit courses taught for industry on-site.

(3) grouping credit courses into a special PC Specialist Recognition Series.

(4) a combination large-group lecture/small-group lab to provide individualized instruction in a cost effective manner.

We would like to share our program development strategy with other colleges who are also seeking to maximize resources.

INTRODUCTION:

The personal computer is a tool of great power for improving instruction and for administrative data processing. The demand for training on personal computers can also result in significant enrollment increases in college courses that use this tool. At Atlantic Community College we first trained our faculty in personal computer use, and then worked to integrate computer usage and training in the classroom.

Although we are a relatively small college of 5500 students, we give 25 sections each semester of an "Introduction to Computers" course which deals with both standard computer applications and also computer literacy and terminology. The phenomenal growth of students in this area led the college to seek to leverage our training expertise to increase our economic resources.

Our aim was to put in place programs that would give Atlantic Community College sustainable economic advantage, what economists call "Monopoly Rents" (Clemons, 1991; Gurbaxani & Whang, 1991) by examining our key competencies and packaging programs in innovative ways. We determined that our key competencies were that we held to high academic standards and that we have strong relationships with the community through advisory councils. We decided to use the advisory councils to assist us in program development.

PROGRAMS:

(1) The Six-Month Computer Programmer Series

The Computer Information Systems advisory council told us that the need in the community was for students who could code RPG programs for the AS/400. We had no AS/400 but did have expertise in RPG III. We

were able to find a compiler that emulates the AS/400 and teach the RPG on a PC. he CIS advisory council expressed a need for mature students that could be trained intensively and be able to enter a Data Processing shop as computer operators but also have the ability to advance to programmer. Our non-credit area had been the source of previous six-month programs such as Word Processing Specialist, but they felt they did not have the expertise to develop a Computer Programmer Specialist six month program. The faculty from the credit area designed the curriculum for this series, starting with Introduction to Computers, Information Systems and Programming, through Programming in Pascal, C, COBOL, and RPG. The final programming project is completed in any language and requires coding a database. The series concludes with a two week internship.

All the courses in this "non-credit" series use the same curriculum as the credit courses of the same name. All students take exams and receive grades. We have completed three of these six-month programs, and are presently in our fourth. The employers are very happy -- we had 15 students in the last group and 26 employers wanted interns. By the end of the internship, 10 of the students had already been placed, despite the high unemployment rate in our area. This project is succeeding because we leveraged our core competencies. We did not lower our academic standards so the college maintains its reputation in the community, yet there has been substantial economic gain to the college. The students are happy because they get college level training in an intensive six-month program. Employers are happy because they have the employees that our area needs. At the last breakfast meeting of the Chief Information Officers of the largest segment of our employers, the casino industry, one of the members commented that Atlantic Community College was producing the workers needed who "really knew something."

(2) Credit Courses Given On Site

Another project is the training of worker on site. We will provide college courses to any employer in the area who meets the minimum requirement for students, which in our case is 15. The course taught is the regular credit course. In some cases, a full time faculty member is the teacher. If a adjunct teaches the course, that adjunct is provided with a faculty mentor, who visits the course site, and calls the adjunct at least every other week to discuss teaching strategy and answer questions. We have standard syllabi for our courses so we can be assured that while course is not offered on our campus it does meet our standards.

(3) PC Specialist Recognition Series

Our many sections of "Introduction to Computers", taught on campus, are attended mostly by non-majors. These students, often non-degree seeking, had questions about what course to take after the initial experience. To give these students both direction and a goal we established a four course "PC Specialist" Professional Series. A student takes Introduction to Computes, Information Systems and Programming, Microcomputer Applications Using Lotus 1-2-3, and Microcomputer Applications Using d-Base IV. They then receive a letter of recognition from the college. This has been very successful, and often after students complete the 12 credits they decide to stay at the college and attain a degree.

(4) Large-Group Lecture, Small-Group Lab

Atlantic Community College, like most community colleges, prides itself on the personalized attention given to each student. Our "Introduction to Computes" classes are limited to the number of PCs per lab, which is usually 20. This course has a lecture component as well as a lab component, and the teachers soon found themselves moving demonstration computers from classroom to classroom for the lecture. The obvious solution was to use large lecture sections, in which the equipment was in place and could be shown to 60 students at a time. We did not, however, want the students to be in contact with the instructor only for a large-group lecture, so we coupled this with small-group labs. One lab section is linked to 3 lab sections, so that 60 students are served by faculty whose load is six credit hours, as opposed to nine. Faculty were willing to do this because they felt

that the large lecture was better for the students and because they still got to know the students in the small group sections.

CONCLUSION:

Each of the above four programs has continuous evaluation. If our standards are lowered, we stop the program, regardless of how financially advantageous it is. For example, we experimented with giving one week three credit courses. The students loved them. We kept the same curriculum for "Introduction to Computers", but instead of 15 weeks the students spent from 9 am to 5:30 pm every day for a week completing the material. After two years, we decided, despite the financial advantage to the college, to discontinue this one week format. We determined that the students did not learn the same amount of material as the standard 15 week course. We know there is a market for intensive courses, and are investigating the "open entry, open exit" format. Nevertheless, it is important to remember that programs provide sustainable economic advantage only if the core competencies are not compromised. In tough economic times it is difficult to give up the money making programs, even when the data tell us that students are not learning as they should. We have found, however, that keeping high standards is the best guarantee of economic stability, and because of this continue in our programs to "earn monopoly rents."

REFERENCES:

Clemons, E. (1991). Corporate strategies for information technology: a resource-based approach. Computer, 24(11), 23-32.

Gurbaxami, V. and Whang, S. (1991). The impact of information systems on organizations and markets. Communications of the ACM, 34(1), 59-73.

A Workshop for Faculty about Internet Services

Ernest C. Ackermann
Department of Computer Science
Mary Washington College
Fredericksburg, VA 24401
ackermann@s850.mwc.edu

Faculty, administrators, and students are getting greater access to the Internet. Most students in academic institutions have known about and used computers for a good part of their time spent in school. This contrasts with the backgrounds of many faculty and administrators. They received a good deal of their formal academic training without the benefits of desktop computers or without thinking about using a computer as an ordinary tool in their work. Many can still remember the pleasant shock of knowing that they would have a computer, usually on the desk, that would be available for word-processing and managing data. The academic community has gone through that period so that now computers are truly an ordinary or commonplace tool for those working in professional settings. The availability of computers and the existence of networks have spawned other phenomena and made other services available. It is becoming commonplace to think of a computer as a communications tool and a resource for information in addition to its other uses. Thus it is appropriate to provide training in the form of workshops concerning accessing the Internet for faculty and administrators.

The Internet provides possibilities for communication and sources of information and as such is a valuable, effective and constructive resource. Potential users and other users need training on the use and availability of the services, tools, and information available. Here we will describe some services available over Internet as they are presented to faculty and administrators in a workshop at a college of the liberal arts and sciences. We conduct a day-long, (10 AM - 4 PM), workshop on accessing the Internet. It consists of presentations, examples, and supervised exercises. Most of the participants are regular users of personal computers. The participants get hands-on experience by using PC's to access on our systems that has a version of UNIX as the operating system. Some of the tools they use were developed locally as Bourne shell scripts using UNIX commands and others are available on the Internet. The workshop is meant to serve as an introduction to accessing and using the services available on the Internet. It is organized around the following topics.

What is the Internet?
Electronic Mail
Interest/Discussion Groups
Telnet
Ftp

We will address each of those topics in what follows. In each case we shall first discuss methods or tools that are available and then give a portion of the workshop presentation.

What is the Internet?

Most participants are aware that it is possible to access computer systems in a variety of locations. They're generally not aware of the number or scope of systems accessible through Internet. A brief presentation about Internet and some services available seems to be a good way to start the workshop. Here is an example of what may be presented in the workshop.

THE INTERNET

The Internet is a network of networks, consisting of over 5000 networks and over 600,000 computer systems. It allows personal communication between users of computer systems on the Internet. This is most often done through electronic mail. It allows access to and the exchange of information between a wide variety of users in commercial, educational, military, governmental, and other organizations. This is often done through interest

groups or discussion groups. It is also possible to obtain information from an assortment of sources. These include university libraries in the US and other countries and the Library of Congress; a variety of research laboratories; databases of information dealing with topics as disparate as art history, extragalactic data, literature, and molecular biology; archives of software for many different types of computer systems (MSDOS, Macintosh, UNIX); and individuals engaged in research and applications at several levels. These latter are available by logging in to remote computer systems using telnet and by transferring files among computer systems using ftp.

Electronic Mail

Everyone in such a workshop is aware that networked computers can be used for electronic mail. This portion of the workshop covers an introduction to electronic mail addresses, communicating with users on networks that are not part of Internet (such as BITNET and Compuserve), and a tutorial on the use of **elm**, the user interface and user agent available locally. The tutorial covers issues related to reading, sending, replying to, saving, printing, and deleting mail. Participants go through exercises to cover those topics. This gives the participants some familiarity with the computer system they will be using as well as a sense of using a computer network for communication.

Much of the material in this section is dependent on the system being used for mail. The details will differ if the system is a UNIX, MS-DOS, Macintosh, VMS, or some other system. The following is some of the material presented in the workshop concerning electronic mail address on Internet and some other networks. Eventually some users will come across addresses that don't fit cleanly into the address forms described below. There are several variations that are best addressed on an individual basis, and not appropriate for discussion at this level.

ELECTRONIC MAIL

Electronic mail is one of the most used services on networked computer systems and Internet. It is a convenient way to communicate with others. While it doesn't provide the some of the personal contact associated with face-to-face meetings or telephone conversations, you can send and read messages at your convenience. All that's needed is access to a computer connected to the Internet and knowing another person's address.

Addresses on Internet are of the form **local_address@domain_address**. For example in the address ackermann@s850.mwc.edu, the local_address part is ackermann and the domain_address is s850.mwc.edu. The local_address portion is often the login-id of a user. In some cases the local_address is more complicated, but that is the concern of the software that handles the delivery of the mail in a domain. The domain_address portion indicates the Internet name associated with a specific organization. Its form is two or more fields each separated from the other by a period (as in s850.mwc.edu). The last portion of the domain_address often indicates the type of organization. For example, educational organizations have .edu as the last part of the domain_address. Some others used are .com for commercial organizations and .mil for military organizations. The domain_address can also indicate that an address is outside of the United States. The domain_address coco.edu.uk indicates that the address is routed to the United Kingdom.

It is also possible to communicate with users on some networks different than Internet. Two examples are BITNET and Compuserve. BITNET addresses typically do not contain periods in the part of the address following the character @. If an address on BITNET is friend@fifvm it can be reached from Internet by using the address friend@fifvm.bitnet. Compuserve addresses are related to the Compuserve account number. If one has an account on Compuserve and the account number is 12345,1234 the address from an Internet site is 1234,1234@compuserve.com.

Interest/Discussion Groups

This portion of the workshop explains electronic interest groups and discussion groups to the participants. It also provides them some exposure to searching for groups they may wish to join, and experience in joining such a group. The points covered are

(1) identifying or choosing a group
(2) the address used to join a group
(3) the address used to contribute to a group
(4) services are available from groups
(5) unsubscribing or leaving a group.

A list of interest groups available on Internet is kept and update monthly at dartcms1.dartmouth.edu. It is available by anonymous ftp and questions concerning the list can be addressed to David Avery, System manager for Dartmouth College, David.Avery@dartmouth.edu.

Here is some material from the file siglists/read.me which describes the form of the list.

> Dartmouth maintains a merged list of the LISTSERV and manually maintained lists on Bitnet and the Interest Group lists on the Internet. It is a single file with one line for each mailing list. The list has been updated monthly for the last year and has a stable format. Each line consists of seven fields, delimited by tabs. The list contains information about approximately 2,200 lists. The fields are:

	Field Size
Category (Computing, Science, Humanities, etc)	1
List name	3-11
"Mail to" E-mail address for submitting mail to the list for distribution	8-80
"Command to" E-mail address for subscribing, signing off and other administratrivia	8-80
One line description of the list subject	8-60
Name and E-mail address for the owner of the list	0-80
Long description of the list	0-450

The ftp site also contains programs written in Pascal to allow searching an viewing the list. At our site we use a UNIX shell script, named igroup, to allow users to search and view the list. This allows users to search for a group based on a string anywhere in the description of the groups, and it also provides them with information about subscribing to a group and where to send contributions to a group. Once someone subscribes or joins they usually receive information concerning services available and how to leave or unsubscribe from the group. Some examples concerning the use of the local tool, igroup, and what users might expect after they submit a "subscription" to join a group are presented.

The workshop has included a presentation by a faculty member in the humanities concerning the value and use of these interest or discussion groups. Participants, faculty members themselves and novices can relate well to this presentation. After that participants engage in some hands-on experience in choosing, and subscribing to a group.

The following is an example of some of the material that is distributed in connection with this part of the workshop.

There are over 2,000 interest-groups or discussion lists available to users on Internet. These cover a large range of topics. Some are rather esoteric, such as WELDS for discussions on computer modelling welds, and others are much less specialized such as NUTWORKS which is a digest of the Usenet group rec.humor.funny. These are particularly useful to users who wish to discuss issues with a larger group than may be present at one

institution, who wish to find resources related to their interests or classwork, or who wish to be informed of some of the current activity within a discipline or sub-discipline. Participating in such a group may mean simply reading interesting mail from others or engaging in electronic conversations with other members of a group. The groups also provide an excellent opportunity to forge contacts with others with common interests. Some of the opportunity comes from the fact that communication is on an equal footing and is often non-intrusive. Questions of availability and status are often left behind when users engage in electronic mail or interest groups. These groups also provide a means to receive current information about conferences, research efforts, and special occasions.

INTEREST GROUPS

The command **igroup** displays information on list-servers or interest groups. The information is displayed on the screen if no option is used. Using the option **-m** mails the information to you.

Use it as

igroup pattern

where pattern is a string of characters.

For example:

igroup foreign
igroup foreign lang

If pattern foreign or foreign lang is contained in the description of the list-server/interest group then the description(s) are displayed on the screen.

Another example using -m:

igroup -m foreign
igroup -m foreign lang

If pattern foreign or foreign lang is contained in the description of the list-server/interest group then the description(s) are displayed on the screen.

The following show a portion of the result of using

igroup -m artificial intelligence

==========

LIST NAME: AI
DESCRIPTION:
Expert Systems and Education

To subscribe to AI send the message

SUB AI your full name

to AI-Ed-Request@SUN.COM

Note: the message should only contain the line above.

For example: SUB AI Mary H. Washington

220

The address for submitting items to the list is:
AI-ED@SUN.COM

The email address of the owner of the list is:
J.R. Prohaska <prohaska@SUN.COM>

Other Information:
Discussions related to the application of artificial intelligence to education. This includes material on intelligent computer assisted instruction (ICAI) or intelligent tutoring systems (ITS), interactive encyclopedias, intelligent information retrieval for educational purposes, and psychological and cognitive science models of learning, problem solving, & teaching that can be applied to education. Issues related to teaching AI are welcome. Topics may also include evaluation of tutoring systems.

==========

LIST NAME: Medicine
DESCRIPTION:
Artificial Intelligence & Medicine

To subscribe to Medicine send the message

SUB Medicine your full name

to ai-medicine-request@vuse.vanderbilt.edu

Note: the message should only contain the line above.

For example: SUB Medicine Mary H. Washington

The address for submitting items to the list is:
AI-MEDICINE@VUSE.VANDERBILT.EDU
The email address of the owner of the list is:
Serdar Uckun, MD <serdar@vuse.vanderbilt.edu>
Other Information:
Computer scientists and engineers with interest in biomedical and clinical research, and of physicians with interest in medical informatics. AI in Medicine is a broad subject area which encompasses almost all research areas in artificial intelligence. For the purposes of this mailing list, AI in Medicine may be defined as "computed-based medical decision support" or "computer-assisted medical decision making".

==========

[Additional material is presented in the workshop concerning the form of a message for subscribing to a group and the response one would expect form a group if it is managed by a listserver. We also discuss some of the listserv commands. This material is omitted here for the sake of brevity.]

Telnet

The next focus of the workshop is accessing information and services available on other computer systems by using the commands telnet or tn3270. In addition to discussing the syntax of these commands and using these commands, the workshop informs the participants about telnet accessible sites that may be of interest to them. Accessing libraries for bibliographic searches readily attracts faculty and a portion of the workshop is devoted to that. There are other activities and sources of information that are available in the same manner. The software "hytelnet", authored by Peter Scott (scott@sklib.usask.ca, Acquisitions Department, University of Saskatchewan) with UNIX and VMS software by Earl Fogel (Computing Services, University of Saskatchewan, fogel@sask.usask.ca), provides a hypertext interface to a variety of telnet accessible sites. The user needs only to select a library, freenet, campus information system, or some other site and the software invokes telnet or tn3270. In addition to this hytelnet provides a glossary of network terms and other information about telnet, hytelnet, and software used on various library systems. Hytelnet is available by anonymous ftp from access.usask.ca. Here is a portion of the documentation that comes with hytelnet.

> Hytelnet is a general hypertext browser, packaged with Peter Scott's database of Internet-accessible telnet sites. Hytelnet lists libraries, Campus-Wide Information Systems, Freenets and more. Hytelnet is available for Unix, VMS and IBM PC systems.
> ....
> The program notices telnet commands embedded in the files, and will execute them for the user, if desired. TN3270 commands can be executed as well, if this program is available on your system.

Users seem to adapt very well to using hytelnet for telnet access and for searching for information on the Internet. It also provides access to enough material and sources to keep someone busy for quite some time. For example there are approximately 300 libraries listed and over 100 other sites. Here is some of the information we provide in the workshop.

USING TELNET AND TN3270

The command **telnet** or the command **tn3270** may be used to access libraries for bibliographic searches as well as additional sites for other types of information. These include sites which contain weather forecasts, Campus Information systems, various bulletin boards, access to decisions of the Supreme Court, and many others. Both telnet and tn3270 are variations of the same command. Each allows a user to initiate a remote-login to a computer system, and is used as

telnet internet-address
tn3270 internet-address

for example,

telnet Library.bu.edu
tn3270 brownvm.brown.edu

If you use **telnet** then in the majority of cases you must be using a terminal that emulates a VT100 terminal. This will most likely be the situation if you are using a modem connected to a personal computer. The command **tn3270** allows your terminal/pc to emulate an IBM 3270 terminal. The two are not identical and you should follow the instructions for a particular computer system. However, **telnet** is most common.

When you want to exit a session you must remember the following.

To exit a telnet session press **CTRL-]** (*hold the CTRL key, press the] key, and release both*)

You will get the prompt telnet>

At that point enter **quit**

To exit a tn3270 session press **CTRL-c** (*hold the CTRL key, press the c key, and release both*)

You will get the prompt tn3270>

At this point enter **quit**

HYTELNET

hytelnet is a program that provides information on the use of telnet and internet addresses for a wide variety of information. Here is a description provided by the developers of hytelnet.

* HYTELNET is designed to assist you in reaching all of the INTERNET-accessible libraries, Freenets, CWISs, Library BBSs, & other information sites by Telnet.

* HYTELNET is designed specifically for users who have access to the Internet and the Telnet program via modem, serial line, or direct network connection.

Enter **hytelnet** in response to a $ prompt after you log in and you will see the menu below. Use the cursor keys to move around in hytelnet. The up and down ar ows move you from one highlighted link to another.

Welcome to HYTELNET version 5.0
Last Update: January 21, 1992

What is HYTELNET? <WHATIS>

Telnet-accessible library catalogs <SITES1>

Other telnet-accessible sites <SITES2>

Internet Glossary <GLOSSARY>

Cataloging systems <SYS000>

Understanding Telnet <TELNET>

Key-stroke commands <HELP>

HYTELNET 5.0 was written by Peter Scott,
U of Saskatchewan Libraries, Saskatoon, Sask, Canada. 1992
Unix & VMS software by Earl Fogel, Computing Services, U of S 1992

Use the cursor keys to move around in Hytelnet. The up and down arrows move you from one highlighted link to another. Use the right arrow (or <return>) to follow a link, and use the left arrow to come back. You can also use hjkl (vi's cursor keys).

Here is a quick command summary which can be found by using the cursor keys to highlight <HELP> and the pressing the key marked Enter.

HELP!

MOVEMENT:	Down arrow -	Select next Topic
	Up arrow -	Select previous Topic
	Right arrow -	Jump to next Topic
	Left arrow -	Return to previous Topic
	hjkl (vi cursor keys) work as well	
SCROLLING:	+ (or space) -	Scroll down to next page
	- (or b) -	Scroll up to previous page
OTHER:	? -	Help (this screen)
	m -	Return to main screen
	q -	Quit Hytelnet

*** Press the left arrow key to leave HELP ***

Ftp

We lastly come to the notion of transferring files between computer systems on the Internet. We discuss the use of the command ftp. This seems to be the most difficult of the topics covered in the workshop. Beginning users soon feel comfortable at some level with electronic mail and interest/discussion groups. Using hytelnet its relatively easy for them to browse and connect to other computer systems. However, transferring files, copying files, from other systems is in a different category. First of all they need to know which files they are interested in retrieving. The use and availability of archie makes that task easier. Second, the user interface to ftp is cryptic and designed for someone familiar with working with files. Not many novice users fall into this category. Finally, there are several variations on the directory structure of systems on the Internet(UNIX, TOPS20, etc) and various compression methods used for files available by anonymous ftp. It gets confusing very quickly for users at this level. We present the users with a brief introduction to ftp as well as provide an example of its use. A few of the participants are ready for this, but most aren't. However, they will be when they have more experience or confidence.

In the past we've used the ftp-list put together by Jon Granose to help locate appropriate files on Internet. More recently it has been possible to use the various ways to access the archie anonymous ftp databases for this purpose. Archie is accessible by electronic mail (archie@nic.sura.net), by telnet (telnet nic.sura.net, login-name archie), or as a client querying the database. The latter method generally gives the quickest return of information. If one does a good deal of searching for files on Internet then its worth the time to learn the appropriate syntax for either of these forms.

The following is a portion of the material discussed at the workshop. In addition to the ftp example shown there we also include and discuss in more detail other examples.

FTP

ftp is the file transfer program that uses the ARPA standard File Transfer Protocol (FTP). By using this program you can transfer files among UNIX and non-UNIX network hosts. The simple syntax of the command is

ftp internet-address

for example,

ftp wuarchive.wustl.edu

The actions of **ftp** are based on a client-server model. The remote host is the server and the local user, the one who issued the ftp command, is the client. In the example above the FTP server is wuarchive.wustl.edu. We may send commands to the server and data is then passed to the client. If the server allows, it is possible to send data or files to the server as well. Commands that the user issues to the server usually include

dir	list the files in the current directory,
cd	change directory,
get	copy a file from the remote computer to yours,
binary	switch to binary mode for transferring binary files, and
ascii	switch to ascii mode which is the default mode.

There are a number of sites on Internet that allow any Internet user to become a client and request that files be transferred from the server to the local client. These are known as anonymous-ftp site. An example is wuarchive.wustl.edu. When ftp makes a connection with the remote site and a login-name is requested enter **anonymous**. You will then be prompted for a password. First try to enter your internet address, e.g., bobo@s850.mwc.edu. Most sites would like that as the password. Just a few require that you enter the password **guest**.

Here is an example of an ftp session with wuarchive.wustl.edu. The file README is transferred to the user's directory. Often a file named README contains information that a user should know.

ftp wuarchive.wustl.edu
User: <-- enter anonymous
Password: <-- give your email address
get README
quit

One accesses an anonymous ftp site to either retrieve files or send files to the site. The most common activity is to retrieve or "get" files. The files may be in text form (ASCII) or in binary form. The latter is useful for transferring files that are in some compressed format, contain executable programs (for MSDOS, Macintosh, etc), or represent a archive or collection of other files. Before retrieving a file in binary form enter the ftp command binary.

One way to search for files that may be retrieved using anonymous ftp is by using a service named **archie**. In this case you send mail to archie@nic.sura.net requesting the names of files or directories that contain a pattern you specify. For example to request a list of sites that contain files or directories that contain the pattern "ethics" do the following:

elm archie@nic.sura.net
Subject: **prog ethics**
Copies to:
Now enter a message whose first line is **quit**
Anything else in the message will be ignored.

For more information on archie, send archie@nic.sura.net a message that contains the word help on a line by itself, or enter **man archie-doc**. Before leaving this topic there is a quicker way to probe the data base associated with archie. If you enter the command

BEST COPY AVAILABLE

archie ftp.help

then you will get a quick reply which includes the following.

```
Host ftp.uni-kl.de
   Location: /pub/internet
        FILE -rw-r--r--      7701  Mar 27 14:32  ftp.help

Host pilot.njin.net
   Location: /pub/ftp-list
        FILE -rw-r--r--      7701  Jan  3 15:24  ftp.help

Host pprg.eece.unm.edu
   Location: /pub/ftp-list
        FILE -rw-r--r--      7701  Feb 27 19:39  ftp.help
```

You could then retrieve the file ftp.help from pilot.njin.net by entering

```
ftp pilot.njin.net
Name:  anonymous
Password:       your e-mail address
cd pub/ftp-list
get ftp.help
quit
```

Conclusion

The Internet is a means for communication and a resource for a great deal of information. In order to realize the potential of this network of networks users need to receive training on communicating with others, identifying sources, and retrieving information. This paper described an introductory workshop for faculty and administrators. As such it describes some of the tools and methods necessary to begin a more complete use of the Internet. For the sake of brevity some of the portions of the presentation were omitted. All materials used in the workshop may be obtained upon request from the author.

Educating the faculty and staff to computer use: A working model

Donald Armel
Robert Barger
Eastern Illinois University
School of Technology
Charleston, IL 61920
cfdja@eiu.edu

Abstract

Great computer systems and networks are nice to have, but do no good if the users are not properly introduced and supported for effective operation. This presentation will explain how one university has approached the problem of training and support. Eastern Illinois University created a new faculty committee called the Computer Resource Faculty (CRF). These persons act as liaisons between Academic Computing and the general faculty, fulfilling their charge independent of each other and without prescribed duties.

Introduction

Eastern Illinois University's progression from teaching programming through computer science to the use of computers for instructors and instruction is probably very similar to many institutions' experience. The latest enhancement has been the installation of a campus wide network. One of the essential aspects of the installation is the training and support of the general faculty.

This presentation will describe how EIU has approached the problem of training and support by creating a new faculty committee called the Computer Resource Faculty (CRF). The CRF was developed in response to expanded computer use in all aspects of the educational experience. The computing environment is described to provide a reference of similarity to other institutions. The responsibilities and structure of the CRF are outlined, as well as including examples of their activities.

Progression of computing as EIU

Eastern Illinois is a comprehensive state university with 11,300 students. It grants degrees in 83 Bachelor's degrees and has 21 Masters degrees. Physically, the campus is compact for a school its size with only 9 major academic buildings.

Eastern began its educational experience with computing by teaching programming on its local mainframe, using COBOL, FORTRAN and BASIC. In 1974 EIU expanded its use by participating in the Mid-Illinois Computer Cooperative (MICC), which was the sharing of distributed computing services with 8 other universities. The MICC services expanded the resources for teaching programming using two labs having a total of 30 terminals plus dial-up access.

MICC existed until 1987 when EIU's governing board (Board of Governors) took over the services in the form of the Educational Computing Network (ECN) for its five universities. ECN provided new services for students and faculty by adding e-mail, bulletin boards, Bitnet connections, simulation packages, statistical software and manufacturing software. Later an IBM mainframe was added for additional programming and SAS.

The Educational Computing Network provided faculty with the first services for electronic communication, shared printing, file serving and bulletin board interest groups. This laid the foundation for the campus network, by developing some experienced faculty using these services, and by illustrating a need for expanded access.

Current Computing Environment

1600 desktop computers
11 computer labs
1 unix server for math
2 mainframes
Campus network (Uniform Campus-wide Area Network, UCAN)
    Over 2000 connection boxes
    600 currently connected (300 faculty)
    1 unix server for faculty
    8 dos servers
    4 shared printers

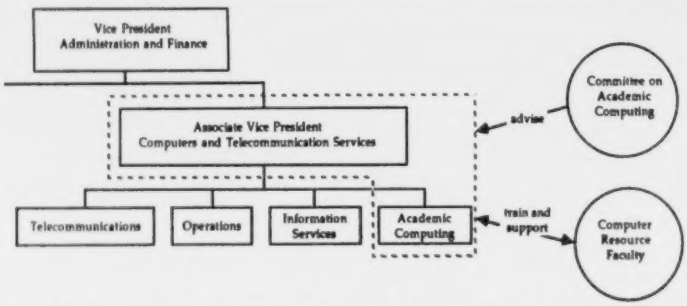
Network Services

E-mail
Bulletin board service with Clarinet news
Internet services
File sharing
Shared printing
DOS server programs
    WordPerfect
    Quattro Pro
    PageMaker
CD ROM searches (9 Data bases)
Library catalog searches
Journal abstract searches
Student record access
Run programs sorting student records
Online advising
Plato

The network was first available at the beginning of 1990, and it is easy to see from the above list that to the novice and non computer user the networks potential could look bewildering. Before the installation of the campus network, the ECN was the only network available and the ECN staff, along with EIU's Academic Computing staff, provided the documentation and local training and support. When the UCAN was developed, it became clear that existing staff were too few to handle the potential 600 faculty needing training and support.

One Institution's Approach:

In 1990, Eastern Illinois University established a standing committee known as the Computer Resource Faculty (CRF). This group was jointly appointed by the Provost and the Associate Vice President for Computer and Telecommun- ications Services. For several years there has been the Committee on Academic Computing, a group of Deans and Chairs who advise on the hardware development and services provided for all academic computing concerns. As the network was installed it was understood that the systems success depended on a will supported group of users. The CRF was conceived and proposed to have at least one faculty member was appointed from each of the major academic units of the University to make a committee of 11 members. The relationship of these committee to the computing area are illustrated below.



The CRF act as liaisons between Academic Computing and the general faculty, fulfilling their charge independent of each other and without prescribed duties. Some of the faculty seemed to feel that the staff of Academic Computing, being computer professionals, were too technically oriented to be able to comfortably interface with the faculty's academic problems. To service all teaching and non teaching faculty at least one CRF member would be selected from within each college. Representation across the campus is divided as follows:

College/Unit	members
Liberal Arts & Sciences	3
Education	1
Fine Arts	1
HEPR	1
Business	1
Applied Sciences	1
Continuing Education	1
Testing Services	1
Library	1

The CRF member should be accessible so their college's faculty could come with computer-related questions and problems. This college-based CRF member would either answer the faculty member's questions directly or would refer them to a specific staff person in one of the Academic Computing offices. This would reduce the number of requests for information coming directly from the University's faculty to Academic Computing. It would also provide a more local, approachable, "one-of-us" type persons in each college who could serve as a point of first approach for college faculty. Also, as time permitted, the CRF member could provide computer-related training sessions or workshops for members of their college.

At present, the CRF members are a service group, performing their duties without a budget or credit-unit recognition as part of their annual workload. They do, however, receive up to $500.00 in annual travel grants from their college deans.

The CRF model is still a very loosely organized group. Their charter is only to act as a liaison with no specific set of duties or guidelines describing how that is to be accomplished. Intentional or not, the vagueness does allow extreme flexibility in servicing each College's needs. Each CRF creates their own objectives and provide the services requested for their computing demands. Listed below are examples of the activities performed by several of the CRF in response to a campus questionnaire:

- workshops specifically for the use of electronic mail as a means of speedy interaction with colleagues on campus, as well as around the world

- workshops specifically for the use of electronic bulletin boards as a way of discussing current concerns and receiving information useful to the faculty's teaching and/or research interests

- contacts on an individual basis with faculty members to alert them to computer-based resources related to their particular disciplinary interests

- tutoring of individual faculty members (including Deans) to improve their facility with computers and to deal with their problems and questions in "hands-on" computer sessions
- liaison work with the University administration and with Academic Computing (usually handled through the medium of CRF meetings) to assure a smooth two-way flow of information regarding computer related matters, including the provision of resources to the various colleges
- demonstration sessions to illustrate the information access resources from the Library: CD-ROM, card catalog, abstract searches, and software downloading
- address student groups interested in the network services for group communication and educational resources
- developing instructional materials to make the unfamiliar unix operating system, e-mail and bulletin board services easy to learn and appear friendly
- instructed on the use of the dos faculty servers, shared printing and the related software
- instructed persons on how to access student records, enrollment figures, setting registration flags and other advising functions
- solved general computing problems like: installing dos 5.0, helping with computer graphics, made recommendation on pc purchases.

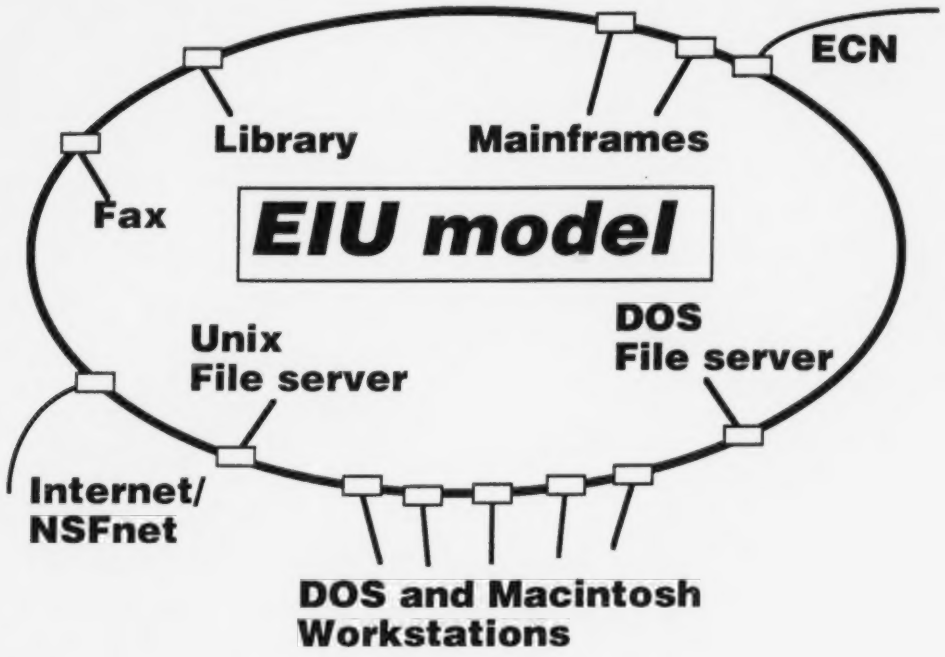
Summary

Key Implementation Concepts

1. Minimal specific duties facilitation flexibility of activities to match each Colleges' faculty needs.
2. Faculty trained by peers who understand their needs and can empathize with beginner users.
3. Frequent group meeting with Academic Computing.
4. Train the trainer.

CRF Needs

1. Financial resources (to the committee and individual CRF) for developing training materials, purchasing educational software, inviting speakers, etc.
2. Recognition of activities as part of workload.
3. Release time to organize and conduct training.
4. Department leaders' support as user role models.



DOS File server
Shared applications:
- WordPerfect
- QuattroPro
- SAS

Shared Postscript
- laser printer

File sharing

Internet
Telnet
- conversation mode

FTP
- file transfer protocol

Service subscription
- Illinet - library db.
- Weather
- 66 international libraries

Mainframe IBM
Student records
Academic programming
- & program submission

Online transcripts
Campus calendar
Electornic filing

Library Services
Online catalog ILLINET
Journal abstracts CARL
CD-ROMs
- business
- education
- science
- social studies

Gov't. documents
Shareware

ECN
Educational Computing Network
- Unix
  - E-mail
  - Bulletin Borad
  - Internet & Bitnet
- Mainframe
  - SPSS, SAS
  - Programming

A wide area network
made up of 5 universities
- Chicago State
- Eastern Illinois
- Govenors State
- Northeastern Illinois
- Western Illinois

Unix File server
E-mail
Bulletin board
File sharing
Shared printer
Online phone directory
Internet

Helping Adults Learn
Planning and Implementing Effective Inservice Programs
at a Small Liberal Arts College

Elizabeth Lancaster Braswell
Director of Computer Resources
Mount Olive College, Mount Olive, NC

Introduction

Helping faculty at small liberal arts colleges remain abreast of advances in academic computing presents a problem for most most colleges where resources are limited. Faculty have varying levels of skills and interest in computing and have heavy workloads. Nevertheless, it is imperative that colleges find a way to upgrade faculty skills in computing and to help faculty integrate computing into instruction across the curriculum. Accreditation agencies mandate that computer resources be available to faculty and students and that computing be incorporated into instruction. The literature reports that careful planning and attention to the design of inservice programs facilitate effective adult learning. Colleges have a number of options related to preparing faculty to use computers in instruction. The faculty development opportunities in computing at Mount Olive College, Mount Olive, NC are designed and implemented by the Director of Computing Services.

Programming Processes for Adult Learners

Research suggests that programming for adults involves consideration of more than the content that will be presented to learners. Administrative support, needs analysis, and collaborative planning to meet institutional and individual goals must be addressed. Organizations must provide time necessary for employees to participate in learning programs; individuals must balance the stress of participation with their ongoing work, personal, and family responsibilities.

Mandated participation in programs which consider only the organization's need for employees to develop specific skills risks resentment and lack of cooperation. Attempts at extrinsic motivation and/or "pressure from the boss" cannot dictate learning. Effective learning depends on adults' coming to understand that it is in their own best interest to participate. Their motivation may be related to the organization's sanctions and rewards, but adults must find value in participation. Adult learners may be willing to adopt the organization's goals for the program, but they also want an opportunity to incorporate personal learning goals. Collaborative planning fosters a win-win situation where the organization and individuals both meet their needs.

According to Hutson (1981) three domains are involved in the development of successful inservice programs: conceptual, procedural, and substantive. The Conceptual Domain involves consideration of the type of inservice which include intensive training, workshops, and/or sabbaticals. Consideration of administrative elements such as the balance of power and authority, the allocation of resources, collaborative planning, motivation of learners, reward/sanctions constitute the Procedural Domain. Content and the delivery processes are the focus of the Substantive Domain. Content should be a blend of theory and practice. Hutson stresses that inservice programs should model the following good teaching methods:

1. Encourage active learning.
2. Use self-instructional methods.
3. Allow great freedom of choice.
4. Involve demonstrations.
5. Provide for supervised trials and feedback.
6. Adapt to real life conditions of adults.

Knowles (1970) reports that adult learning focuses on solving a problem rather than on subject matter acquisition within a formal learning situation such as a college course for credit. Self-direction is fundamental; adults want to share in the planning and contribute from their varied experiences. Readiness to learn is based on developmental tasks. A child's role is to go to school; adult roles are related to physical, unlearned drives and to needs and aspirations. Sources for adult tasks may be from the values and concerns of the learner, from the environment and society, and from the physical needs related to the human body. Adults enter learning activities to deal with their problems and, therefore seek immediacy of application. Adults want to understand how a learning project will help meet their individual goals and help solve their problems.

Variations in learning effectiveness is to be expected according to Bloom (1976). He indicates that if instruction is presented to a group of twenty to seventy learners, some will learn effectively while others will find it ineffective. He suggests that three interdependent variables account for variation in learning and asserts that if these variables are attended to, instruction can be approximately error-free:

1. The extent to which the student has already learned the basic prerequisites to the learning to be accomplished.
2. The extent to which the student is (or can become) motivated to engage in the learning process.
3. The extent to which the instruction is appropriate for the learner.

Tyler has proposed a system whereby the elements of learning can be systematically examined to identify those processes which facilitate learning. In an interview recorded by Carter, Tyler indicates that if you want to know whether or not people are learning, you must "look at what they're doing, not simply at the product" (Carter, 1976: 49). Mujie (1981) delineates the elements of Tyler's learning system and provides extensive evidence that attention to and compliance with those elements have indeed facilitated intentional learning in a variety of settings and situations. According to Tyler (Carter, 1979), an effective learning system includes various elements that must be considered in designing adult learning programs:

1. Clear objectives.
2. Motivation.
3. Learning tasks.
4. Opportunities for practice.
5. Reinforcement and feedback.
6. Transfer and generalization.
7. Reinforcement and feedback in transfer situation.

Element 1: Clear Objectives

According to Tyler (1985), before learning can take place, learners must clearly understand what they are to learn, what is the end goal. Tyler indicates that learners must know what they are to do; they must become conscious of the criteria to look for. Carter (1987: 17-18) adds, the "clarity of objectives and the degree to which the learners understand or come to understand them" influences the extent to which persons choose to engage in learning activities. Behavior change is not limited only to overt behavior, but also includes "thinking, and feeling, and acting" (Carter, 1976: 31). Learning involves acquiring skills, knowledge, and attitudes that learners did not have previously, but which enable "them to guide and teach themselves" (Carter, 1976: 20). Objectives must deal with what learners will do, not with what the teacher will do. Learning objectives must be clearly delineated, understood, and adopted by the learner. Ultimately, what adults will learn must be a decision made by the adults themselves. They will only engage in activities that meet a perceived need.

Element 2: Motivation

Carter (1987: 17) asks: Do the learners have ample reason, in their own interest, to engage in the activities that are being provided? This question has to do with the clarity of the objectives and the degree of

motivation of the learners. Unless, for their own purposes, they see reason to engage in the learning activities, useful learning is not likely to occur. Tyler (Carter, 1976: 9) states that adults must want to try before they can learn. Although teachers frequently discuss ways to motivate students, in fact, motivation is not something that can be applied from outside the learner. Motivation occurs within potential learners when they recognize a gap in their learning and sense the need to fill it.

Element 3: Learning Tasks

Selection of appropriate learning tasks is essential for intentional learning to occur, according to Tyler. Implementation involves translating the objectives into learning activities that help the learner come to understand that which is to be learned. While education has long been defined in terms of what the teacher does to induce learning, looking at what happens within a learner is more appropriate. The emphasis must be on what happens to the learner. Tyler warns that teachers too often plan for their own activities, what they will do in the learning situation, when their concern should be to develop means whereby the learners can practice the desired behaviors. Learning activities must first begin with simple concepts and then move to the more complicated concepts. The learning activities must be designed in steps that enable the learner to recognize progress and feel satisfaction. If the tasks seem either too easy or too difficult, learners will not be motivated to continue.

Element 4: Opportunities for Practice

Learners must have adequate opportunities to engage themselves in learning tasks, to practice that which they are trying to learn. Learning involves functioning differently; this of necessity requires adequate time addressing the learning task. The ability to restate a given theory is vastly different from the ability to incorporate that theory into practice. Not only must trainers address acquisition of skills, but also they must provide for adequate practice of those skills. Opportunities for practice must continue until the skill is error-free. This condition, of necessity, requires that the facilitator be available to give immediate feedback.

Element 5: Reinforcement and Feedback

Learning is facilitated when learners have guidance and feedback from those who are helping them regarding how well they are performing the tasks they are trying to do. Learners need assistance from a facilitator who can suggest ways to correct problems and also let them know which things they are doing correctly. The facilitator in these ways can reinforce the learning process and can help learners feel confidence in their progress. Helping learners recognize their accomplishments contributes to their expectation of the probability of success. This, in turn, reinforces their motivation to learn and intensifies the desire to continue the activity. According to Tyler (1985), skills, knowledge, and attitudes develop slowly, and the incorporation of those skills into practice, likewise, occurs over time. The learning period may continue long after formal instruction has ended.

Element 6: Transfer of Learning

According to Tyler (Carter, 1976: 13), learners first apply a concept to a specific class of phenomenon then recognize that the concept can be applied to other classes as well. Not only must learners be able to demonstrate desired behaviors within the learning context, but also they must be able to generalize those behaviors so that they can demonstrate them in other situations. To insure transfer, learners must be able to practice in varied situations and settings. The fact that learners can demonstrate success within a learning situation does not necessarily mean that they will be able to reproduce the behaviors in other situations.

Element 7: Reinforcement and Feedback in the Transfer Situation

As learners leave the training situation and try to transfer their learning to actual situations that occur within their own work, they still need guidance and feedback on their progress. They need continued support and interaction with one who can assist when questions arise as they integrate their new learning into practice. Ongoing support land feedback must be available to learners.

Reported guidelines from educational research and recommendations from the literature became the framework for designing faculty development in computer technology at Mount Olive College.

Faculty Development at Mount Olive College

Mount Olive College is a four-year private liberal arts college serving the North Carolina region eastward from Goldsboro, North Carolina, to the Atlantic Coast. The college enrollment is approximately 768. In 1984, the college purchased Apple IIe computers and developed a computer lab for student use. The Business Department offered a computer literacy course which was part of the general education requirements. In that same year, the Department of Languages and Literature became the first department to integrate computer instruction into the general curriculum. English faculty provided instruction in word processing using AppleWriter and later AppleWorks and required that English assignments be completed using the computer.

In 1989, the college purchased Macintosh computers for use in journalism and advanced writing courses offered by the Department of Languages and Literature. Professors in English and art developed a course in desktop publishing which was team taught. The success of those courses led to the development of a new minor in Professional Communication in the Department of Languages and Literature and a new major, Visual Communication, in the Fine Arts Department. In 1990, the college purchased IBM computers for use in the Business Department's new major, Computer Information Systems.

Although some demonstrations and workshops were provided for faculty from time to time, no concentrated effort was directed at faculty use of computers until 1991. Interested professors developed computer skills on their own as computer use became more widespread. New faculty recently out of graduate school brought developed skills with them.

An institutional self-study which was part of the process and recognition of the importance of the computer as a developing technology provided the impetus to address academic computing needs. The accreditation agency mandates that colleges demonstrate integration of computing use across the curriculum and develop written policies related to allocation of computing resources and to student access to computers. Prior to the fall of 1991 Director of Computer Services was appointed and was charged with coordinating academic computing functions. The position is part-time; the Director continues to teach courses in the Department of Languages and Literature.

Beginning in the 1991 fall semester, the Director of Computer Services planned and implemented a comprehensive inservice program for faculty according to recommendations found in inservice literature. Learning opportunities were available using Apple, Macintosh, and IBM computer platforms. The program offers a variety of delivery systems: informal consultation, special topics miniworkshops, regularly scheduled computer courses, and intensive job-related seminars. The Dean and the President strongly supported the program, but participation was voluntary.

Informal Consultation: The Director's role as an informal consultant involved availability to faculty as a resource person. A spacious office with a conference table is provided for the Director to permit small meetins and discussion groups. Activities included Table Talk, weekly informal discussions over coffee, in the Office of Computer Services every Tuesday from 9:30 till 10:30. These informal gatherings followed special faculty seminars discussed below. These sessions had no specific agenda but included software and hardware support and trouble shooting, purchase information, assistance to self-directed faculty engaged in discovery learning, and demonstrations.

Special Topic Mini-Workshops: Student tutors periodcally presented mini-workshops in word processing in the computer labs for interested persons. Although originally planned for students, workshops were open to faculty. Topics focused on skills needed to complete English assignments: introduction to the computer, basic word processing functions such as entering, editing, and printing text, cut and paste. Other workshops addressed advanced and special functions for word processing such as research paper format, page numbering, title pages, and reference pages. General interest workshops included topics such as letter and memo forms, resumes and

job application letters. Student lab tutor/technicians working in the computer labs during afternoons, evenings, and weekends provided general assistance upon request.

Computer-Related Classes: As a fringe benefit, faculty may take one tuition-free course each semester. Participation in regularly scheduled computer courses did not develop during the fall and spring semesters. However, faculty interest during 1992 summer sessions was so great that a new section especifically for faculty was added to the schedule.

Intensive Computer Seminars: The Director developed and implemented a series of intensive computer seminars for faculty and staff during the fall and spring semesters. In the fall, three seminars were planned to meet in the computer lab on Tuesdays during the 8:00 -9:30 class period. In the spring, two seminars, each for six weeks, were planned.

Using reported guidelines for effective inservice, the Director conducted a needs assessment and interest survey. Faculty were invited to review the developing proposal and to comment and make suggestions. The seminar schedule and content emerged from these efforts. Faculty who registered for seminars participated in establishing specific learning goals which incorporated not only administrative goals, but also individual professional and personal goals. In general, adults' developmental tasks related to career development are based on age and professional experiences. Young adults want to develop basic professional competencies; middle aged adults seek advancement and responsibilities. Successful older professionals, generally more self-confident and secure in their positions, address personal values and concerns related to their jobs.

Participation in faculty seminars was voluntary, but was strongly supported by the administration. The Dean, who had little computer experience, participated in the first two seminars planned for beginners. Administrative rewards included consideration of computer competences as part of performance evalutions and merit pay increases. A certificate of achievement was presented to participants and was made part of personnel records upon completion of a seminar. Personal rewards were self satisfaction and an increase in self esteem.

For efficient learning to occur, adults must feel safe to take risks. They fear of failure and embarrassment. Faculty are especially concerned that their lack of skills not be exposed to students. The Director carefully developed a "safe" learning environment which provided for social interaction, cooperative learning, and a "buddy plan" support system. Participants developed a group identity and special comaradrie with each other.

Tyler's Learning System provided guidelines for designing learning activities for the seminars. During the first class, the Director provided a general introduction to the seminar topic and surveyed participants' entering skill level; participants reached consensus on agendas for remaining sessions. In an immediate hands-on activity, learners developed a simple document and produced a computer printout during the first class. These first printouts created great excitement and a sense of accomplishment. In succeeding classes, the Director stressed ongoing informal evaluation of the seminar process, and faculty were free to modify agenda topics for succeeding weeks.

The Director assumed a role as a facilitator of learning, rather than as expert instructor. She used active learning, practice and feedback, and coaching strategies to assist learners in meeting their personal goals which by this time incorporated administrative goals. Ongoing self evaluationwas encouraged. As participants achieved beginning goals, the Director encouraged them to establish new goals. In general, faculty reported that they learned much more than they had expected to.

The Director provided learning activities based on the learners' level of understanding and immediacy of application in the performance of their jobs. Learning tasks were based on job-related functions for faculty such as developing a test bank, test preparation, correspondence formats, mail merge, syllabi, reading lists, book orders, and course handouts. Agendas remained flexible enough to move quickly through tasks which the learners quickly achieved and to move slowly enough to insert additional activities related to more complicated tasks. Instructional strategies were developed using a modules approach which was the basis for printed agendas. These

included participant "show and tell," questions/answers, discussions, mini-lectures and/or demonstrations. A major part of each class involved hands-on practice with feedback from the facilitator and peers.

Self evaluation and program evaluation have identified strengths of the program design and problem areas that need modification. Informal consultation, special topics miniworkshops, regularly scheduled computer courses will again be available for faculty. Plans for 1992-93 intensive seminars began at the end of the spring semester with a needs assessment and circulation of signup sheets to suggest topics of interest and to register for seminars. The success of the first seminars have created interest in participation. Topics for next year's seminars will include telecommunications, electronic bulletin boards, Internet, integration of computing into instruction, as well as introductions for beginners.

Selected Bibliography

Bloom, Benjamin S. Human Characteristics and School Learning. New York: McGraw-Hill, 1976.

Carter, G. L., Jr. ed. Facilitating Learning with Adults: What Ralph Tyler Says. Madison: Division of Program and Staff Development, University of Wisconsin Extension, 1976.

Hutson, Harry M. "Inservice Best Practices: The Learnings of General Education." Journal of Research and Development in Education 14 (1981): 1-10.

Joyce, Bruce R. and Beverly Showers. "Transfer of Training: The Contribution of 'Coaching." Journal of Education 163 (1981): 163-72.

Joyce, Bruce and Beverly Showers. "The Coaching of Teaching." Educational Leadership 40 (1982): 4-10.

Joyce, Bruce and Beverly Showers. "Improving Inservice Training: The Messages of Research." Educational Leadership 37 (1980): 379-85.

Kidd, J. R. How Adults Learn. Chicago: Follett, 1973.

"Practical Applications of Research: Inservice Education." Newsletter of Phi Delta Kappa's Center of Evaluation, Development, and Research 5 (1983): 1-4.

Tyler, Ralph W. Basic Principles of Curriculum and Instruction. Chicago: The University of Chicago Press, 1949.

- - -. "A Learning System." Lecture. North Carolina State University, Raleigh, NC, 14 Feb 1985.

Developing and Integrating Computer Labs in Non Computer Courses

Francis X. Foley
Department of Business & Economics
5024 Berry College
Rome, GA 30149-5024

ABSTRACT

This paper discusses the considerations and problems inherent in the implementation of a program of integrated computer lab requirements in traditional academic business courses. Consideration is given to who should be involved in the process, how much integration is optimal, what that integration should be and what the process is intended to accomplish. The advantages if integration are discussed along with the particular strategy necessary during the design of integrated courses. Observations are made regarding the use of available popular mainstream application software, as well as the utilization of course or discipline specific applications and/or simulations. Advantages are examined from the perspectives of the faculty and the student.

Berry College is a small, private liberal arts college nestled in the hills of northwest Georgia, located approximately seventy-five miles northwest of Atlanta, with about sixteen hundred fifty undergraduates and one hundred full-time faculty. As with any college these days, especially small private colleges, the resources are limited while the requirement to remain competitive is tremendous.

For some years a primary concern of the college has been the need to provide appropriate computing services to faculty and students. Computing resources were recognized as an indispensable component of the core curriculum as well as an essential part of any comprehensive recruiting effort. Issues such as the availability of and access to public-use computer labs and types of hardware and software on campus continue to become more and more important.

The Computer Utilization Committee was formed in response to the administration's recognition of the differing and complex needs of the various departments. The school faced the prospect of the various departments and administrative offices making costly and potentially incompatible decisions. Additionally, there was the probable expensive duplication of resources and the difficult, if not impossible, problems inherent in trying to share information. These concerns combined with the expected support problems, led to the appointment of a Director of Academic Computing.

Through the office of the Director of Academic Computing, the Computer Utilization Committee is composed of administrators and faculty members from all the schools and chaired by the Director of Academic Computing. The committee represents a cross section of campus computer users, and coordinates faculty training, hardware and software purchases. The Director of Academic Computing also services the software site licenses. Consideration was given to the problems of "power users" vs. "novice users". Faculty members selected for inclusion on the committee are tasked with the responsibility of "mentoring" other faculty members. Those chosen had a high degree of literacy and skills and a commitment to improving the computer literacy of the campus community and the implementation of computer enhanced teaching.

Site licenses were purchased for WordPerfect, as well as a comprehensive site license with Borland Corporation. At the request of a faculty member, approved by his/her Dean, any of the site-licensed programs are provided along with the appropriate instruction. Any instructional books or manuals are purchased by the individual schools.

Being charged with, or participating in, the development of a plan for integrating computers and other computer-related technology into the curriculum is equivalent to being asked to look into a crystal ball and predict the future. Some sources indicate that the life cycle of technology is such that it out-dates itself every two years. Therefore, it was imperative that any recommendations made or programs implemented, be ones which would allow the institution to expand with the new technologies without wasting time, money, or opportunities for student and faculty growth.

The committee has been investigating software that is being used successfully at other academic institutions as well as encouraging their colleagues to examine programs specific to their disciplines. With establishment of a campus-wide computing policy with common goals and objectives, opposition was minimal. Additionally, this process has provided a cohesive plan for managing computing resources on campus in an effective, efficient, and non-threatening manner.

As a member of EDUCOM, the college regularly sends faculty members to the annual convention to learn the latest in instructional technique and product availability. Our vision is to enable students to use the advanced technological tools available in their fields. To realize that vision, specific underlying structures needed to be developed.

At a small liberal arts college, computer literacy might be defined as the ability of students to use the microcomputer to perform the tasks in his/her discipline. In general, computer literacy could be considered to be the ability to operate a microcomputer, have knowledge of the operating system, a word processing package, a spreadsheet, and a data base manager.

One attempt to meet the demands of our constantly changing world has been the increased integration of computers in the educational experience. The College is beginning to offer all incoming Freshman an Introduction to Computers course with emphasis on word processing and spreadsheets. In addition, there are computer courses in the School of Business and in the School of Science and Mathematics. All schools at the college have structured courses that require the use of computers, some requiring only word processing skills. The challenge has been to get more faculty to become involved.

The first step in the process of integrating computers in the educational experience was to provide all faculty members with a computer in their offices and assist them in developing the skills necessary to make use of the currently available software. Group instruction was provided in the form of structured hands-on computer exercises supplemented by one-on-one assistance by the committee members from their respective departments.

The second step has been to encourage faculty to incorporate more computer use in their courses. At present, the primary uses are word processing, spreadsheets and data base managers. In the business statistics courses more use is being made of computer-based statistical processing with the use of MINITAB. Additionally, students are encouraged to prepare all course papers using word processing software and are generally offered the opportunity to submit drafts for review prior to the submission of the final paper.

Unless the instructor is dedicated to the use of microcomputers and finds innovative ways to use them as a basic part of instruction, he/she is apt to subscribe to what appears to be the general consensus today. "My course/discipline already covers so much material that there is no time to incorporate teaching computers." It should not be necessary to give up teaching the subject to teach computers. Proper support by the designated person from The Computer Utilization Committee should provide that service for the faculty member.

We are finding it necessary to insure that the primary computer-oriented faculty member in the department be proficient in the course support package prior to its adoption by the rest of the faculty members. It is difficult to encourage the primary computer-oriented faculty member in a department to spend the hours necessary to improve skills when so much emphasis is placed on research and writing.

It is essential to include faculty members from affected areas in the decision-making process, and demonstrate how computer components in courses can provide an arena where students can be guided through the more difficult concepts at their own speed rather than that required in the classroom.

Examples of commercially available software can be demonstrated to faculty and opinions about their applicability can be solicited. Spreadsheets can be demonstrated for use in accounting, finance and marketing courses. Econometric modeling programs and graph-generating packages plus basic computer accounting packages can be reviewed for their use in specific courses. Personally developed computer-based tests and activities as well as computer based grade books using spreadsheets, can also be demonstrated.

We have found it helpful to demonstrate how microcomputers can offer more flexibility for the instructor and the student. Students, while learning the primary subject, will also improve their skills and learn the value of the proper use of decision-making tools. For most students a computer will be another tool in the performance of their activities.

Microcomputer use by faculty will, to a great extent, depend on the quality and the type of software available in his/her discipline for use in the classroom and in after-class activities. The two primary types of academic software suited to this environment are tutorials and simulations (games), each with their own advantages and disadvantages (for specific use). It helps to remind faculty of the old adage: "different students learn in different ways."

A tutorial provides a way for a faculty member to reinforce their lecture and the text with activities outside the class room. Tutorials do have their problems, however; while they may cover the content, they may not do it in the most attractive and interesting manner. Students will have to be self-motivated to spend that much time in front of the monitor. Many textbook publishers provide excellent tutorials and study guides with built in self-test opportunities.

It is in simulation, however, where we seem to get the most student involvement and response. Simulations allow students to get more involved, make decisions, and examine their effects on the modeled activity so that they become active participants rather than passive observers. Simulations use competition to excite students and give them an incentive to cover the material. A simulation should cover the best features of each lesson to get the students to look at more complex situations. While the use of a simulation will not guarantee a student's interest, it can serve as an exciting and enticingly effective instructional tool. Simulations have the potential of providing the student with a realistic decision-making environment in which various important business concepts can be integratedand reinforced in ways that are very difficult to achieve in the traditional classroom environment.

Microcomputers offer more flexibility for the instructor and the student. Students, while learning a skill, also learn the value of the proper use of decision making tools.

Computer use by faculty is improving, but not without some continuing problems. It takes time for faculty to develop the skills and confidence in the use of both the hardware and the software. There are a limited number of classrooms where computer-assisted instruction can be conducted. As demand increases so will this problem. While there are many excellent programs on the market today it is very difficult for faculty members to find just the right program when so many demand are made on their time (researching, writing, etc.) in addition to their teaching and advising responsibilities. To try to convert everyone at once is a daunting task which can only lead to frustration, disappointment, and resistance. Be methodical, involve those affected by any decisions regarding the implementation of hardware and software, and be patient.

Promoting Computer Literacy Among Faculty and Academic Staff: The Approach at a Small Liberal Arts College

Patrick Kopp
Joan Lescinski, CSJ
Avila College
11901 Wornall Road
Kansas City, MO 64145

The purpose of this paper is to describe how Avila College, a Catholic, four-year, co-educational college with liberal arts and professional programs including graduate programs in Education and Psychology as well as an MBA, has gone about promoting computer literacy and usage among faculty and academic staff on the campus. It has been a multi-year project which is still on-going.

To understand the process the faculty and academic staff are engaged in, some background on the college and its faculty and on computer usage on the campus will be useful. After that the paper will describe the program the college has instituted for computer acquisition and literacy among faculty.

The Avila College faculty, made up of 89 full-time equivalent professors, serves a student body of approximately 1,500. The college computer program really began in 1986 when Avila had 19 Apple IIe microcomputers and a PDP 11/24 minicomputer serving academics (a PDP 11/24 is a minicomputer with about 500,000 bytes of main memory). The one at Avila used 2 disk drives, each capable of holding about 30 million bytes. Twelve of the Apples were kept in one room for teaching purposes which was kept locked unless a class was in session. Six more of the Apples were in the library for students and faculty to use any time the library was open with last Apple in the nursing department. These seven Apples were purchased on a nursing grant. The PDP 11/24 computer was used mainly for programming languages associated with the computer science majors, and the only faculty who had computers were the ones who brought their own.

Very few faculty used the Apples for everyday work because the lab was locked except for classes. Access was nearly impossible since many faculty had their offices in other buildings at a distance from both the library and the computer lab. It was also a problem for faculty to use the library computers. Since this decreased faculty availability to students, the PDP was used by only the computer faculty and students taking computer science classes.

In the summer of 1986, the business chairman purchased two IBM compatible computers and donated them to the school. These computers were 8088 machines with two 5 1/4" floppy drives and no hard drives, and this gave the business secretary and faculty better access to computers and began computer usage on campus.

Next, the nursing department purchased a Macintosh Plus in 1987 on a grant and two IBM compatibles in 1988 on another grant which gave the nursing department a Macintosh for special presentations, an Apple and IBM compatible for office use, and an IBM compatible for faculty use. Still, however, the general faculty did not have any improved access to computers, so interest in computer literacy and usage remained low. Equipment acquisition was a piece-meal, unplanned process at this point.

In 1988, the PDP 11/24 minicomputer was replaced with a Prime EXL-316 minicomputer which was leased because it was capable of running UNIX software, the compiler for the computer science language courses, and MS-DOS software. The college hoped to be able to run WordPerfect, Lotus 1-2-3, and dBASE III PLUS on the mini also. However, compatibility problems developed with the terminals and software because the terminals were DEC VT220's and did not have function keys that many MS-DOS software packages use extensively.

There were also problems with multiple users. For example, WordPerfect would work well for one user. If, however, a second user tried to use WordPerfect, the computer would block the program. The only way to

get WordPerfect to work was to have the executable file located in each user's account, but this caused memory problems.

In 1989 the school was able to come up with the money to purchase an IBM lab. In July of 1989, 12 IBM model 30-286 computers were purchased along with a model 60. The model 60 was used as a file server and the other microcomputers were networked using NOVELL, the leading network software. This was the beginning of integrating computers into the classroom, and this was also the beginning of acquisitions for the library when we purchased an IBM Model for them to provide access to computers when the labs were being used by classes.

In order to get the academic offices computerized, two IBM compatible computers and two IBM model 25 computers were purchased for the other four academic departments: Humanities, Social Sciences, Education, and Natural Sciences. This completed the effort to make available in each of the academic departments at least one computer, generally by the department secretary, but with a small amount of increased usage by the faculty.

When the Prime lease expired in July 1991, the college decided that the programming languages: BASIC, COBOL, Pascal, C, and FORTRAN, could be taught on microcomputers, so instead of renewing the lease, the money from the lease payments was used to purchase Macintosh computers. The Macintosh lab, purchased in January of 1991, consists of 8 IIsi's, 5 IIci's, a full-page color scanner, a film printer, a laserwriter, and 4 two-page monitors. This lab has multiple uses: by the art department for desktop publishing and computer graphics classes; by the computer science department for software application and some programming courses; by the faculty and staff for presentations and general reports; and by the students for papers, research, and, sometimes even for fun. An important point to note here is how the college was able to fund the project. The dean, with the agreement of the president, borrowed the money from the endowment with the stipulation that the academic area would repay the loan of $80,000 over a several-year period at 9% interest. In the current climate of 3-4% interest, this arrangement obviously benefits both the academic area and the college endowment fund.

One of the most significant breakthroughs occurred when the college was awarded a Federal grant in October 1990 that consisted of approximately $600,000.00 to be distributed over a three-year period. Through this grant, microcomputers could be purchased for academic use. During the summer of 1991 seven IBM compatible microcomputers were purchased. Two of these computers were distributed to the dean's office and the library. The other five computers were placed on movable carts and given to the departments. For the first time, faculty had easy access to a computer. As a result a number of faculty began to incorporate computer usage much more systematically into their course planning, preparation, and presentation.

Additional equipment which could interface with computer technology was also purchased such as two PC viewers, devices that project the images from a computer screen onto a wall through an overhead projector. One is available in the computer labs and the other is located with one of the portable computers mentioned above which allows it to be set up and used in classrooms.

In January of this year sixteen IBM compatible computers were purchased for faculty use along with one Macintosh 2si for the library to relieve pressure on students and faculty when the labs were in use. Some means had to be developed to determine who would receive these computers so the academic dean sent out a survey to all faculty. The survey asked if the faculty member would like a computer and, if so, what would he/she use the computer for. One innovative aspect of the survey was that it indicated that the people receiving the computers would be required to train another faculty member when the next batch of computers are purchased. More than twenty faculty responded positively, requesting a computer.

The Academic Computer Advisory Committee, given the task of selecting the recipients from the list of people wanting a computer, decided to allocate the first four to department chairs who did not already have a computer. Eight were given to faculty based on need and lack of access to one of the portable computers. Most of these eight faculty are area coordinators or program directors. Three of the remaining computers were

placed on portable carts and made available in areas where there is a concentration of faculty. This made it possible for more faculty to have access to computers. The last computer was given to the business department secretary because the business department was still using the computers purchased in 1986 which contained no hard drives.

The last year of the grant, which is yet up-coming, will allow the purchase of ten more computers after October 1992. The distribution of these computers will be based on need, accessibility, willingness to learn how to use the computer, and willingness to train another faculty member. When this last group is purchased and distributed, the campus will have a ratio of about one computer for every two faculty, but the ratio is actually higher because many faculty have their own computers on campus.

One point to note is that the art department was not given an IBM compatible computer because these faculty members are Macintosh-oriented. The art department purchased a Macintosh Quadra in March 1992, and their plan is to purchase an additional four Macintosh computers over the next three years.

This summary of the acquisition pattern and rationale for the equipment leads naturally to a consideration of the training given to faculty to enable them to use the hardware and software. The training started in 1988 when faculty were invited to participate in workshops on using the Apple computers, Apple Works and BankStreet Writer, and on using the new Prime computer, but this particular round of training was not very successful for several reasons. First, as mentioned before, most of the faculty did not have computers in their offices or easy access to the computer lab. This made it difficult for the faculty members to practice what they learned. Secondly, the people who already had their own computers did not feel they needed additional training. The ones who attended the workshops were people who were not planning to use the computers. They were attending more for the experience. However, as might be predicted, because these faculty could not practice what they learned, the training proved relatively ineffective.

When the IBM lab was installed in 1989, more training was planned. Since IBM computers were on campus, it was reasonable to assume that faculty would want to learn how to use them. Training began on WordPerfect and expanded to Lotus and dBASE. The same problems arose. Faculty could not be in their offices to answer the phones or be available for students and work in the lab at the same time. A few weeks after a workshop, the faculty member would forget the training.

To compound the problems, the secretarial support staff received no training. When they received their computers, they were given a manual and, in some instances, a workbook. If secretaries had problems, they were expected to solve the problems on their own or, possibly, receive some help from the computer science faculty. This proved to be time-consuming, inefficient, and frustrating.

With departments buying their own computers and software, another problem arose: diversity of software. The nursing department was using Microsoft Works. The business department was using SuperWriter. The rest of the departments were using WordPerfect. There was no way to share forms or documents, especially for faculty working together on a project or committee, or for secretaries producing information needed across departments.

The academic dean and the Computer Advisory Committee, which is made up of a group of interested faculty, made a decision to pick one word processing package and use it in all academic departments. The package chosen was WordPerfect, the most current 5.1 version. The nursing and the business departments are now in the process of converting files over to WordPerfect. This is allowing the standardization of academic forms and information, such as scheduling, book ordering, and so on and allowing the sharing of this information and collaboration among departments and individual faculty. One example of this is the current updating of the handbook for academic advisors. Eight faculty and administrators are involved in the project and are able to share the revisions in process since the entire project has been converted to Word Perfect. In addition, the committee has approved other such packages as Pagemaker, PhotoShop, Lotus, dBase III and SPSS for use by faculty and students. In its on-going work, the committee now routinely reviews requests for new software and upgrades, providing a measure of planned growth in this area for the college. One example of this is the recent

request by the committee that each academic department develop a three-year plan on how to integrate computer technology into the classroom.

All of this increased usage and sharing of information was made possible by a grant written by the dean and others which allowed the academic area to start purchasing microcomputers to be placed into department areas. As mentioned earlier, some of these computers were put on carts and placed in areas easily accessible to the various department faculty. Training sessions were then set up, not only for the new computers, but also for other A/V equipment that was on campus. The dean appointed a committee of faculty, including the director of the computer center, to devise and carry out these training sessions. The syllabi for them were determined as a result of a detailed questionnaire which was sent out to the faculty late in the summer, and the first of these training sessions began in October of 1991.

The first workshop, starting in October 1991, was on the use of the Macintosh color scanner and the development of transparencies on the LaserWriter. Its three sessions attracted twenty-two participants. All faculty were asked to bring something they would like to see on a transparency. As an example, the Radiologic Technology professor brought a book containing MRI photos. Others brought diagrams and documents, and each attender was given a chance to perform a scan. Both the graphics package and the optical character recognition (OCR) package were used. Once the image was scanned and stored in a file, the faculty member was shown how to perform editing. The final document or picture was then printed onto a transparency. The participating faculty were pleased with the ability to enhance their use of transparencies in the classroom.

The second workshop, starting in November 1991, covered the use of the film printer and the production of slides. Two sessions were offered and a total of fourteen faculty and staff participated. Participants were given a demonstration on how to prepare information and then send it to the film printer. Among the topics discussed were types of film, graphic software, and the quality available.

Additionally, a workshop was offered about the same time discussing the use of the camcorder. Faculty were interested in learning how the camcorder could be used in classroom situations. Two sessions were offered with a total of nine participants. At present, we are experiencing an increase in the willingness of faculty to use video-taping as an instructional tool along with other technologies which are more computer-related.

When the sixteen computers were purchased in January 1992, we decided to set them up in one room so that all the faculty receiving a computer could be trained at one time instead of individually. On a day in January, just prior to the beginning the Spring semester, twenty-five faculty and staff were shown the intricacies of the new computer and printer which covered getting on the computer, printing a document, saving, and editing it. The equipment was distributed the following day to each faculty or area. All equipment was in place before the start of the Spring semester so that faculty would be able to use it immediately in the preparation of materials for their Spring classes. This procedure avoided the previous problems mentioned earlier when training and practice had been separated in time.

Once the faculty received their new computers, the idea was to get them started on putting them to use. Hence, a workshop was offered on beginning WordPerfect within two weeks of the beginning of the semester. Three sessions were offered with a total participation of thirteen. Subjects covered included starting up, entering text, editing, retrieving files, saving to disk, printing, and exiting.

An advanced WordPerfect workshop was to be offered approximately three weeks later. The committee sent out a worksheet on other features of Wordperfect that the faculty could implement before the next workshop which included underlining, bolding, text size, and centering. It also included a list of topics that would be covered in the advanced workshop in hopes that this would keep their interest until the advanced sessions started.

The advanced WordPerfect workshop began in mid-February. There were fifteen participants spread over three sessions. The subjects covered included manipulating the format menu such as margins, tabs, page

numbering, headers, footers, fonts, and so on; using spell check and thesaurus; moving and copying text; merging files; creating subdirectories; copying files from one drive to another; retrieving of one document into another; and addressing envelopes. At the end of this session, faculty were urged to practice in WordPerfect. If problems came up, they were to ask the secretaries who, by now, had achieved some proficiency in WordPerfect, or to contact the academic systems manager for assistance.

In early February 1992, a workshop was offered on how to set up and use the PC viewers, and its two sessions attracted seven participants who were taught how to connect the PC viewer to the computer and then disconnect it. An explanation was given on how to adjust the projection for contrast, brightness, and so on. Again, this has now enabled another group of faculty to make use of computers and related equipment in the classroom.

The final workshop this semester covered MS-DOS. Two sessions were offered in March and April which were attended by twelve faculty. Subjects to be covered include: booting dos, dos commands such as dir, format command, copy command, diskcopy command, and deleting files; working with subdirectories which included creating subdirectories, changing to the subdirectory, and removing a subdirectory; and path and autoexec.bat.

Related to faculty training and increased usage of computers was the issue of staff support and training, particularly the academic secretaries. Without the ability of the secretaries to keep pace with the computer facility of the faculty, the efficient use of computers would be greatly impaired. So, at the beginning of the fall semester 1991, the secretaries, in coordination with the dean, started a series of meetings to discuss computer usage. The dean now meets at least once each semester to assess on-going needs and plans for increased computer usage with the academic secretaries. An example of the effectiveness of these meetings is the development of macros. A few of the secretaries had been developing macros in WordPerfect for different forms used on campus. The meetings with the dean have been used to trade macros, develop new macros, and come up with ideas to become more standardized on campus. The college anticipates that this will continue to simplify the exchange of information among faculty, chairpeople, deans, registrar, advisement, and student services.

Plans are being developed for future technology growth. The campus is looking at networking, realizing that it will be a slow process since, as would be typical in institutions the size of Avila, there is not enough money to network the entire campus at one time. What will most likely be done is that small networks will be installed, training will be provided for affected people, and problems will be worked out for each small network as it is installed. One example is the planned networking between the registrar and the departments to facilitate advisement, course scheduling, and degree checks. As money becomes available, more networks will be installed and eventually all networks will be connected through a fiber optics backbone.

With the installation of the networks, faculty will have access to the administrative computer. This will open up avenues into advising, registration, electronic mail, and many other benefits.

An important associated development concerns the up-grading of the library's computer-related facilities where, in recent months, the college has purchased Dialog and Infotrack as well as an on-line catalog system. Computer networking with MCAT for the State of Missouri holdings and OCLC for inter-library loan are increasing the ability and willingness of both faculty and students to use computer on-line search techniques in their research.

This paper indicates that Avila College has not developed into a state-of-the-art computer wonder campus. Rather, it started small, purchasing a few computers as money became available. As in all new ventures, mistakes were made along the way. But mistakes have taught valuable lessons such as the fact that training is effective only if the equipment is available for use immediately after the training. Another is the realization that the purchase of computers and the training of personnel on these computers will be a never-ending task. But the result of these several years of concerted effort on the part of the college has shown that, even with limited resources, it is possible to begin to enable an entire faculty to acquire a significant level of computer literacy.

A Practical Approach to Faculty and Staff Personal Computer Training

Mick Lantis
Purdue University North Central
1401 S. US 421
Westville, IN 46391

Introduction

With the PC Revolution sweeping into every area of academia, there is a surge of interest among faulty, administrators, and support staff to become more productive with their personal computers. While there has been a considerable amount of effort given to acquiring adequate hardware and software, the training element of effective information processing has been largely overlooked. The personnel-training strategies have traditionally fallen along three lines: the traditional "for-credit" courses that the institution offers, the "non-credit" or "continuing education" courses that the institution offers, or in "self-taught" courses--i.e. the personnel desiring training should just read the software's documentation and learn on their own. Unfortunately, none of these resolve the issue of providing adequate computer training.

There are several problems associated with having personnel enroll in for-credit courses. These courses require a significant commitment of time, much of which is devoted to testing and other activities that aren't of major benefit to a person simply wishing to acquire a better understanding of how to increase their productivity with their personal computer. Also, is difficult for faculty to attend classes with individuals who are also their students. The problem associated with having personnel attend continuing-education courses is that the courses are generally lacking in depth, and may not cover the applications that the person is using in their work function. Of course, the problem with expecting personnel to train themselves with user's manuals is that this is basically just a euphemism for ignoring the training issue altogether.

The answer to the dilemma is to develop a computer-training strategy that will provide personnel with relevant and thorough instruction. The following issues should be considered when devising such a strategy.

The Personnel Involved

The same basic training needs exist among faculty, administrators, and support staff. Each group has training needs specific to their job function: faculty are interested in instructional uses of computers, administrators are interested in using computers for management information, and support staff are interested in using computers for information processing applications. However, there is a common thread among all groups, and that is the need for training in system fundamentals, operating systems, and personal productivity applications. These are the primary areas that should be addressed.

One particularly difficult consideration is at what level the training should begin. Since this depends upon the personnel's current abilities, it is a good idea for a pre-training survey (i.e. test) be given to determine an appropriate starting point. Ideally, students could be segregated into different training sessions based upon their abilities; however, as in any venture of this sort, practical considerations of time and facilities may require that students with a variety of abilities be trained together. This is just a complication that the person providing the training must deal with in the classroom. But in order to speed up the training process and ease frustration, the material to be covered, and the assumed level of computer ability should be communicated to the students before they begin the training.

The Training Environment

Concern should be given to both the physical aspects of the training environment, and also to the scheduling and duration of the training courses. The scheduling should be done in such a way as to accommodate the job responsibilities of the personnel being trained. Early morning or early afternoon sessions seem to be the most acceptable. Two-hours is the optimal length for a class period--long enough to cover a substantial amount of material, yet short enough so the student's job responsibilities aren't compromised. While the length of the courses depends on the amount of material to be covered, the best approach to a large amount of material is to split the course up into beginning and advanced courses. Experience has shown that a course meeting for two hours per session, two sessions per week, and two weeks in duration is the optimal amount of time.

The best setting for the training sessions is in a computer classroom that have hardware similar to the equipment that the personnel are using. It is of utmost importance to isolate the students from their work areas, phones, and job responsibilities. The training sessions should be a time where they can concentrate solely on the subject at hand. Since the training time is limited, it is a good idea for the instructor to use a copious amount of handouts and written notes, and not depend upon the personnel to take notes. A data display device for an overhead projector is also an essential piece of equipment.

The Subject Matter to be Covered

The subject matter that the training is to cover should be determined by a survey of what the personnel's needs are. If hardware and software standardization has not been promoted in the organization, this task might become extremely difficult. For example, if each department or organizational unit has purchased their own favorite brand of software, the person in charge of training could be confronted with a half-dozen different brands of word processing packages to provide training for. Generally though, the basics of operating systems are common to everyone.

The following are the results of a typical survey of 25 faculty, staff, and administrative personnel. This information can be used to determine training needs:

Operating Systems	22
Windows	18
Spreadsheets	16
Word Processing	13
Graphics Programs	10
Advanced Operating Systems	9
Database Management	9
Networking	8
Desktop Publishing	7
Programming	4
Integrated Packages	1
Utility Programs	1

The survey indicates that operating systems (DOS and Windows) are of interest to the largest number of people. This isn't surprising in light of the ubiquitous nature of operating systems, but it is surprising that so many people actually want to delve into the complexities of system software. Many of the respondents indicated that they are tired of being at the mercy of system professionals to perform such tasks as fixed disk management, and would like to learn how to maintain their own system. In light of this, an operating systems class should be one of the first training courses offered--and perhaps even made a prerequisite for the other training classes. The other survey results indicate a desire for training in personal productivity tools, with a minor interest in programming and other applications.

Developing User Training/Support: Hints for Success

Jeanne V. Livers
Southeast Missouri State University
One University Plaza
Cape Girardeau, MO 63701

"WANTED: SCIENTIFIC ANALYST/PROGRAMMER, COMPUTER CENTER, SOUTHEAST MISSOURI STATE UNIVERSITY". Scientific Analyst/Programmer? What's that? As I continued reading the ad, the following phrases seemed to stand out from the text:

". . . degree in Computer Science . . ."
". . . teaching experience helpful . . ."
". . . background in statistics . . ."
". . . experience with microcomputer applications . . ."
". . . good communication skills . . ."

I could hardly believe what I was reading. The job description was perfect for someone such as myself. My qualifications were as follows:

- I had graduated from this same University with a Bachelor of Science degree with a major in Computer Science.
- I had also received a Bachelor of Science in Secondary Education. I had taught for four years in public schools, and for six years at the University.
- My major in Education was Mathematics, and I had taken two statistics courses.
- While teaching Computer Literacy and Information Systems at the University, I had gained knowledge in the area of popular PC applications.
- When I was working toward my Education degree, my minor was English.

Never had I dreamed that such a wide variety of interests could all fall into place under a single job description. But such was the case because in June, 1990, I began my duties as a Scientific Analyst/ Programmer. I soon found out that I was only the second staff member of the Academic Computing sector of the Computer Center. For at least ten years prior, my supervisor had single- handedly performed all of my new duties, in addition to his other countless responsibilities. I knew I could easily become quite useful in this department, and I was ready for the challenge!

"THE WAY WE WERE"

One of the first items on my agenda was to examine the current status of computer training and support for faculty and staff. I wanted to see how mainframe and PC usage had developed since I had first graduated in 1980. What I found was somewhat disturbing. I could quickly identify four main problem areas that demanded my attention.

First, microcomputers were becoming increasingly popular campus- wide, but understanding of basic PC concepts was lacking in many users. I got the distinct impression that some offices had purchased the systems because "everyone else was buying them". Once the microcomputers arrived, the users were simply told to familiarize themselves with this new technology without any formal training. This resulted in many an office worker learning how to load a word processor, type a letter or memo, print it, and save it. And that was all they could do! They sometimes had no real feel for what happens when a program is loaded into memory and executed. Many of them had never been exposed to other popular PC packages such as electronic spreadsheets and databases. Some individuals consistently "lost" files because the data would be saved in a subdirectory unknown to the user. Too many people had become dependent on menu systems that allowed them to use a few applications with no understanding of even the simplest DOS concepts. No one explained to these individuals how to back up data or prevent virus infections. How could a given user (possibly a victim of computer anxiety) be expected to learn all of these concepts on-the-job while performing all of his/her other duties?

A second difficulty involved the apparent decrease in popularity of the University's mainframe (at least with respect to the usage I had seen as a student and member of the faculty). The ability to send and receive electronic messages was ignored by much of the University community. Hence, too much paper was being wasted by the generation of brief memos and announcements that were being sent to each employee on campus. Due to a lack of understanding, future access to BITNET and the Internet (two major international computer networks) was not viewed campus-wide as a major advancement for the University. Although the mainframe would forever serve the University in an administrative role (i.e., record-keeping, payroll, etc.), I could see another whole side of the system that was not being used to its potential.

The fact that new faculty were never really exposed to Academic Computing presented another dilemma. Certain departments had taken the initiative to tell their colleagues about the computerized test-scoring and survey frequency distribution services. But I could see how a new faculty member might easily get the wrong impression of the Center due to the fact that our computing resources and research assistance seemed so limited. Many teachers were unaware of the fact that the mainframe allowed them to use SAS (Statistical Analysis System), SPSS (Statistical Package for the Social Sciences), Minitab, and Statpak, all popular statistical packages. Some sort of publicity was needed to inform the faculty about what Academic Computing had to offer.

A final problem was associated with a lack of communication between the Computer Center and the rest of the University community. The last issue of the Computer Center Newsletter had been printed almost ten years earlier. Needless to say, much had happened in the Center since that last newsletter hit the press. Some publication was needed to keep the faculty and staff abreast of all Center developments.

"MAKING A DIFFERENCE"

How could my qualifications help me to find solutions to these four problems? I used my background in Education, PC applications, Computer Science/Information Systems, Mathematics, and English to develop and conduct a series of workshops to meet the varied needs of the University. In addition, I became the new editor of the Computer Center Newsletter, which was re-instated after its ten- year absence.

To get a feel for what the University community wanted to learn, interest surveys were sent to all employees. We were not surprised to find that a large number of respondents were interested in every topic listed on the survey. So the following workshops were designed:

* Introduction to Microcomputing
* Introduction to PC-DOS
* Intermediate PC-DOS
* Introduction to WordPerfect
* Intermediate WordPerfect
* Introduction to Lotus 1-2-3

250

* Introduction to MUSIC (Multi-User System for Interactive Computing, University mainframe)
* Introduction to E-mail
* Introduction to SAS
* Introduction to SPSS
* Introduction to TELNET/FTP (Internet access)

These two-hour, hands-on sessions are free to all faculty and staff. Each workshop has a maximum class size of fifteen, so pre- registration is required. The introductory courses are taught on a level that true beginners can understand. Thus far, the workshops have been quite successful, and plans are being made to offer the more popular seminars at least once each semester.

For the special needs of the academics, a tour of the Computer Center is now a vital part of new faculty orientation. Our new "Academic Computing at Southeast Missouri State University" brochures are distributed so that it is clear what is available on the University mainframe. The monthly newsletter has been well- received by the University. System changes, workshop announcements, "Q & A", and interesting network news articles are included in this publication. The newsletter is automatically sent to all deans, directors, chairpersons, and vice-presidents, but over 120 other University employees have submitted personal subscriptions. We are currently designing more brochures to help us in our endeavor to keep our users up-to-date.

"IN CONCLUSION . . ."

Technological changes happen so rapidly that, a year from now, some of what our campus is doing could become obsolete. But we in Academic Computing, in striving to adapt accordingly, have made a commitment to ensure that our faculty and staff remain informed users of technology.

How to Support Computer Labs for 5000 students with a Staff of One

Kathryn McCubbin
Christopher Newport College
Newport News, VA 23606

How to Expand Student Lab Facilities in a Period of Reduced Budgets

Abstract

The directors/managers of most college computer facilities are facing a challenge, perhaps for the first time in their professional careers, that threatens the ability of their institutions to provide the technological resources necessary to prepare students for employment. There are ways to respond to this challenge. The problems created by the economic condition of higher education are addressed, namely, acquiring additional space and equipment, adding technical personnel, and minimizing the negative impact of new regulations and end user dissatisfaction.

I. Background

The College is the youngest four-year college in the Commonwealth of Virginia and was originally part of the College of William and Mary. Established as a two-year college, the College became a four-year, baccalaureate institution in 1971 and, in July of 1977, became totally independent of The College of William and Mary. The College began offering graduate programs in July, 1991.

Christopher Newport serves a culturally multifaceted community in a rapidly growing urban area where the high technology of NASA Langley Research Center, the Continuous Electron Beam Accelerator Facility (CEBAF), and their associated supporting industries are juxtaposed among more traditional heavy industries such as Newport News Shipbuilding, and the even more traditional harvesting culture of watermen and farmers. Four military bases on the peninsula bring in a continuous flux of young people of diverse backgrounds from across the nation. the College attracts predominantly blue collar, low to middle income, working class students. Many are the first in their families to attend any college. Often the College students work full time during the day or evening, and attend school in off hours as best they can.

As of the 1991 fall semester, the College's head count enrollment of 5034 students resulted in a full-time equivalent population of 3,506. This includes 41 students enrolled in the College's first graduate program. The College draws its students primarily from Virginia Planning District 23. For the period 1985 to 1991 the fall semester head count increased from 4151 to 5034, or 21.3%.

As shown below, the mean age of the College's students is 26.5 years. It is estimated that fewer than half of the College's students are of traditional college age. Seventy-six per-cent (76%) are degree seeking and 24% are unclassified. About 17% of all course enrollments are in the evening classes (5:30 p.m. to 10:00 p.m.).

Fall, 1990 Student Enrollment Distributions:

Female - 59%, Male - 41%	Mean age - 26.5 years
In-state - 90%	Out-of-State - 10%
Black - 12%	American Indian - 0.5%
Hispanic - 2%	Asian - 2%
Foreign students - 0.5%	White - 82.0%

The College has made a significant investment in personal computers and has begun moving toward connectivity. There exists a partial backbone connecting Gosnold Hall with McMurran Hall using ethernet over fiber and a 56 Kbps dedicated telephone line connecting the campus to CEBAF (Central Electronic Beam Accelerator Facility). CEBAF serves as a gateway to the regional network, VERnet, which in turn is connected to INTERnet thus giving the College access to any computer in the world which is linked to INTERnet.

Local Area Networks.

There are several local area networks (LANs) operating on campus: two DEC LANs, three NOVELL LANs, a SUN LAN, and two AppleTalk LANs. Electronic mail and file transfer facilities are available in rudimentary form for administration system users. For these users electronic data sharing is generally satisfied by downloading data onto floppy disks which are compatible with a variety of computer platforms. For academic system users there is a full complement of electronic mail and data sharing facilities.

College Library.

The Virginia Tech Library System is installed on a Hewlett-Packard 3000/935 and computer-based cataloguing, reserves, on line catalog and circulation modules have been implemented.

Microcomputers.

The College has an inventory of 483 microcomputers, of which over 120 are based on obsolete technology. There are 282 printers throughout the campus of which 190 are dot matrix, 77 are laser printers and 15 are ink or bubble jet printers.

Major Network Servers and Workstations.

The college has 3 network servers. A Digital Equipment Corporations (DEC) Server 5500 together with 30 model 5100 workstations provide primary support for the academic programs of the college. A SUN SparcServer 2 and SUN 3/280 together with 18 SUN Sparc IPC workstations provide additional support for the Applied Physics and Computer Science department.

Minicomputers.

There are two Hewlett-Packard 3000 series minicomputers. The model 950 supports administrative computing using software developed in house and by the third party vendors COCO, SRN, and BITECH. The model 935 provides support for the library using the VTLS library system.

Supercomputers.

The Applied Physics and Computer Science department houses a MASPAR massively parallel processor from MASPAR Inc. The Single Instruction Multiple Data (SMID) machine provides 2048 processors and uses a Micro Vax 2 as the front end.

Outside Computer Lines.

The College has 3 dedicated data lines: Two to state agencies in Richmond and one to VERnet, the state's subnet of INTERnet. The lines to Richmond, running at 4800 BAUD, support administrative transactions with the Departments of Training & Personnel, Accounts, Taxation, and Planning and Budget. The VERnet connection is a 56 Kbps line connecting the college's computer center to CEBAF in Newport News. This line is attached to the campus backbone network using Proteon routers.

Networks.

The existing campus backbone links the academic computers in McMurran and Gosnold Halls. The network use ethernet over fiber optic cable. Adequate fiber has been installed to migrate to FDDI. Local area networks in McMurran and Gosnold are attached to the backbone.

Voice Transmission.

The College currently receives all of its voice communication management from the Virginia Department of Information Technology. The college's internal phone system is a centrally maintained CENTREX system. Long distance service is provided through SCATS. The college uses C&P Telephone Company for all communication service and repair. Telephones are maintained using a replacement disposal approach. The current voice system is using the campus wiring system at 99% of capacity. The college does not use voice over data or ISDN technologies.

Student Computer Labs.

The college provides 7 microcomputer labs for use by its students. McMurran Hall houses 3 labs that include IBM PC, Apple MacIntosh, Apple Iie, and Digital Equipment Corporation workstations. The Administration building contains the School of Business networked lab of DOS compatible microcomputers. Labs in Gosnold Hall for students in the department of Applied Physics and Computer Science provides DOS compatible microcomputers, Apple MacIntosh computers, and SUN workstations. Smith Hall provides two Apple Iie labs, one for the Mathematics Department and one for the Education Department. A music laboratory consisting of a synthesizer and computerized mixing studio which supports music composition and a CD and Videodisk performance library is currently being installed in McMurran Hall. The college has one computer-based classroom and three lecture halls in which some advanced-technology presentation equipment is available.

Information Technology Personnel

1 Director of Information Technology Services
1 Telecommunications Systems Supervisor
3 Programmers
1 Full time Computer Maintenance Person
1 Part-time Computer Maintenance Person
2 Computer Operators
1 Part-time Secretary
2 Decentralized Programmers
1 Decentralized Telecommunications Engineer

II. Addressing the Problems Presented by Current Budget Constraints

The Space Problem

The rapid rate of growth of the College has filled all available office and classroom space. Providing additional computer lab facilities results in dislodging faculty or staff. Conference rooms, storage areas, and other areas of convenience have long since been converted to areas of need. In spite of these facts the lab facilities continue to grow. Some additional space is acquired by the rental of temporary trailers for student labs. The lack of water and other amenities does that would preclude their use as office facilities does not detract from their utility in a lab environment. Additional lab space is also acquired by installing workstations in traditional classrooms that the room may be used as an electronic classroom, a lab, or a room for lectures. Painting and refurbishing of the labs is accomplished through the efforts of volunteers.

The Equipment Problem

Success in obtaining additional computer lab space presents the next problem: finding the resources to populate the labs with equipment. The establishment of local area networks has been an effective solution to this problem. Obsolete equipment can be used on a local area network but give the functionality of state-of-the-art workstations. As a fail safe for the computer lab these networks, redundancy is provided by installing same software on each of them. Any lab can act as back up support in the event of equipment failure and satisfy the periodic fluctuation in student demand that arises at various times during the academic term. Connection of these local area networked labs to the campus backbone greatly enhances the delivery of computer resources at a very small increase in variable maintenance costs. The fixed cost of adding nodes to the fiber optics backbone is small enough to be considered as a budget line item because existing conduit can be used to carry the fiber optics cable. Thus one node can be added at a time over a sufficiently long period so that no extraordinary funding is required.

High priority is place on administrative personnel to work with faculty in writing grant proposals that involve the procurement of computer lab equipment. Long range planning identifies the requirements of individual organizational units and by integrating and coordinating the requirements of all offices, economies of scale can be obtained which result in establishing a strong case in the competition for existing budget funds.

The Personnel Problem

The state of Virginia has been under a salary adjustment freeze and new position moratorium for three years. Finding ways to satisfy growth under this constraint is difficult at best. However, some positive responses to these conditions can be taken. Part of the evaluation of the faculty involves service to the College. Such service includes being the first line of problem identification and correction for PCs on the campus. Departmental and School units have provided this service which cuts down significantly on the time technical staff take to respond to trouble reports. The College is fortunate in that the local community is populated with retired military who often seek college degrees after retirement. Many "chronologically enhanced" students have extensive electronics and computer skills. The salaried technical staff is augemented with these students (at student wages) to perform much of our PC and data line maintenance. In addition, maximum use is made of automated operator equipment such as unattended backup which frees up scarce salaried personnel. End-user technical support positions are encouraged so that production requests can be satisfied by end- users at the end-user site.

The Regulatory Problem

The commonwealth of Virginia entertains a highly centralized control over purchasing and information technology activities in state agencies. These can be obstructive unless there is an appreciation on both parts as to the rational of their requests. Attendance at regular meetings of the controlling regulatory bodies provides forewarning of new regulation and changes to regulations. Such forewarning allows the College to anticipate problems and accommodate the new regulations into their routine activities.

The End User Problem

The student computer labs present one of the more visible interfaces between the technical support staff and the faculty. What is the responsibility of technical staff in the education of students in the lab? At the College students assistants are employed to support the student computer labs. These student assistants check students in and out of the labs, keep the equipment operating, replenish paper for the printers, report defective equipment, keep the labs clean and answer simple, straight-forward questions. They generally do not have the requisite knowledge and skills to instruct student users in the wide variety of software available in the labs. They refer students with insufficient computer skills to their instructors. This policy is published at the beginning of

each semester. However, many faculty still feel the skill level of the student assistants should be such as to augment classroom instruction. Daily communication with the faculty through personal visits and e-mail, a monthly newsletter, and an annual evaluation survey help to minimize interpersonal friction and foster understanding between the faculty and technical support staff.

Effective Use of Student Employees In A Microcomputer Lab
An Alternate Two-Class System

Michael J. Payne
Department of Computer Technology
Purdue University
West Lafayette, IN 47907-1421
Payne@Purccum

Abstract

There are many problems today with using student employees to supervise microcomputer labs. One such problem is how effective can one use the student resources available. Last year, I proposed a two-class system to my department at my university. This two class system involved hiring student lab assistants at two wage levels. The more experience students would work at night and be paid at a higher wage scale. The less experienced students would work during the day when lab classes were in sessions and be paid at a lower wage scale.

However, due to budget constraints, I was unable set up such a system. Instead, I set up an even better two-class system of student lab assistants. This version of the two-class system involved the use of both volunteer and paid student employees to supervise the computer labs. The paid student assistants work during the night when there are no lab classes in session. The volunteer student assistants work during the day when lab classes are in session. This system is new and has been in use since the beginning of the Fall 1991 Semester. It has been very successful and has saved my department over $2,000.00 in the student lab assistant budget for the Fall 1991 Semester. This paper will discuss further this alternate two-class system and show how it can help solve some of the problems of the effective use of student resources while also saving money. It also will discuss other advantages and disadvantages of this two-class system both to the department and to the student.

INTRODUCTION

A major responsibility for a lab manager of student computer labs is to maintain appropriate supervision of laboratory facilities within the constraints of a limited budget. Since many students are anxious to gain computing experience outside the classroom, they can be used to oversee the facilities during peak usage times. However, scheduling seasoned students along with new students as lab assistants can be a nightmare for lab managers trying to maintain quality lab facilities while at the same time trying to provide opportunities for students. Still, if this is not done correctly, one can also waste valuable resources.

I noticed this problem reoccurring over the last three years, while managing the labs in my department. Trained consultants were assigned to supervise the labs when normal lab classes were in session. Normally during these laboratory classes, there was an instructor in the lab to answer student questions. Since trained student employees were hired to help students with software problems (answer student questions), there was an unnecessary duplication of effort. When these instructors were asked if the consultants were needed, they suggested that the consultants were helpful for changing paper, fixing the printers, rebooting PCs, etc. By scheduling seasoned student consultants when no classes are in session (evenings and weekends) and new consultants when classes are in session to help professors with menial tasks, the instructor, the consultants, the students and the lab manager all benefit.

But the question still remains, can I solve this problem without creating another problem in the budget area, especially since, my lab assistant budget gets little or no increase each year? I think so, depending on your approach to implementing the two classes. In the next section, I will discuss the possible alternatives and then make my recommendation.

ALTERNATIVES

Again, there are several alternatives to supervising a computer lab as follows. One could use faculty to oversee the labs at all times. This is not possible because of the expense. One could also only man the lab with student consultants when there are no classes in session. This is not a good choice because of the support needed by the classes instructors. Still another option is that of using of unpaid volunteer assistants to oversee the lab. My recommendation is that of using both paid and unpaid lab assistants and that is the alternative I will discuss next.

SOLUTION

My solution to the problem of supervising computer labs while at the same time providing the best opportunity for students is to that of using both unpaid and paid student lab assistants as follows:

Unpaid Volunteer Lab Assistants

This group will do manual work such as adding paper to the printers, fixing jammed printers, rebooting PCs, etc.. This group will only work the lab when there are lab classes in session.

The requirements for a student hiring into this group is as follows:

1) A student in a computing major with/without any prior computing major.

2) The student would have to attend a one to two hour training session on handing the following situations:

   a) Starting printer, changing paper, changing toner cartridges, fixing jammed printers, etc.

   b) Starting up and booting PCs, rebooting PCs, etc.

   c) Handling and reporting other hardware problems.

3) The student could not be on academic probation

4) The student would have to sign a volunteer lab assistant agreement accepting certain responsibilities. (Copy enclosed)

Paid Student Lab Assistants

This group will do the same duties as the unpaid group. In addition, they will help students with software problems, etc.. They will work in the labs during times when there are no lab classes in session. They will be responsible for opening and/or closing the lab depending on whether it is a weekday or the weekend. They will be responsible for the lab security and equipment during the evening and weekend hours. Lastly, they will be paid for the hours they work.

The requirements for hiring into this group are as follows:

1) An incoming computer major that has:

   a) Previous lab experience if had not completed the courses stated in b).

b) Completed at least one course in programming and PCs including the study of DOS.

c) Have respectable grades in their computer major courses and not be on academic probation..

2) Attend one to two training sessions on the following:

a) Starting printer, changing paper, changing toner cartridges, etc.

b) Starting up and booting PCs, rebooting PCs, etc.

c) Opening and closing the labs when no faculty are around.

d) What software you are required and responsible for answering questions on.

e) Lab equipment security.

f) How to handle and report student, software, hardware and other problems in the lab.

The benefits to this approach are as follows:

1. I can provide more students with the opportunity of additional learning experiences.

2. I can hire incoming freshman as unpaid volunteers.

   These students can be trained in one to two-day training sessions on handling hardware problems.

   The students will then gain valuable computer experience, and have related experience they can put on their resume.

3. I agree to provide them with a letter of reference from me as the lab manager if they keep a clean work record.

4. I can provide these volunteer lab assistants an opportunity to advance to paid lab assistants after finishing certain required courses in our curriculum.

   Student incentives would include a fully paid position, more flexible hours and higher level of experience for their resume.

   Another incentive would be that of being able to (partially) finance their education,

The primary disadvantage is that of making sure the volunteer lab assistants show up for their assigned work hours. I get around this by making all volunteers sign an agreement before they will be hired. If they do not follow that agreement, they can be fired. Even if they are not fired, they can be passed over for promotion to a paid position.

Another slight disadvantage that could drive up costs is that of using paid help to substitute for unpaid help. I tell all paid lab assistants that if they substitute for volunteer lab assistants during the day when lab classes are in session they do so for free. I can do this is because I only use volunteer lab assistants during the times when lab classes are in session.

The primary advantage is that of reduced cost to run a computer lab. I also found that I can even keep the lab open more hours for less money. These advantages are supported in examples illustrated in the next section.

EXAMPLES

To better show how the use of both paid and unpaid lab assistants will save a substantial amount on ones lab assistants budget, I am enclosing the following examples to compare the cost of running the lab with paid student lab assistants (Example 1) to the cost of running it with both paid and unpaid student lab assistants (Example 2). I have also included an example with the lab being open even more hours to show that it even costs less than with just using paid student lab consultants. The comparison will be over a one semester time period as follows:

Examples 1 and 2

Weekly hours for first 15 weeks (Total of 74 hours/week)

Monday through Thursday	9:30 A.M. to 11:30 P.M. (56 hours)
Friday	9:30 A.M. to 5:30 P.M. (8 hours)
Sunday	1:30 P.M. to 11:30 P.M. (10 hours)

Finals Week (no classes in session) (Total of 59 hours/week)

Monday through Thursday	9:30 A.M to 10:30 P.M. (52 hours)
Friday	9:30 A.M. to 4:30 P.M. (7 hours)

Example 3

Weekly hours for first 15 weeks (Total of 84 hours/week)

Monday through Thursday	9:30 A.M. to 11:30 P.M. (56 hours)
Friday	9:30 A.M. to 5:30 P.M. (8 hours)
Saturday	1:30 P.M. to 11:30 P.M. (10 hours)
Sunday	1:30 P.M. to 11:30 P.M. (10 hours)

Finals Week (no classes in session) (Total of 59 hours/week)

Monday through Thursday	9:30 A.M to 10:30 P.M. (52 hours)
Friday	9:30 A.M. to 4:30 P.M. (7 hours)

Also, the examples will be based on the following wage rates and specific work schedules:

Example 1: Paid lab consultants are paid at a minimum wage rate of $4.25/hour and work all the hours.

Example 2: Volunteer lab consultants are paid nothing. Paid lab consultants are paid at a minimum wage rate of $4.25/hour. The actual work schedule is as follows:

Volunteer Lab Consultants	Monday through Friday 9:30 A.M. to 4:30 P.M.
Paid Lab Consultants	Monday through Thursday 4:30 P.M. to 11:30 P.M.

Friday 4:30 P.M. to 5:30 P.M.

Sunday 1:30 P.M. to 11:30 P.M.

Finals Weeks Hours

Example 3: Volunteer lab consultants are paid nothing. Paid lab consultants are paid at a minimum wage rate of $4.25/hour. The actual work schedule is as follows:

Volunteer Lab Consultants	Monday through Friday 9:30 A.M. to 4:30 P.M.
Paid Lab Consultants	Monday through Thursday 4:30 P.M. to 11:30 P.M.
	Friday 4:30 P.M. to 5:30 P.M.
	Saturday 1:30 P.M. to 11:30 P.M.
	Sunday 1:30 P.M. to 11:30 P.M.
	Finals Week Hours

SUMMARY

	Example 1	Example 2	Example 3
Weekly Payroll	$314.50	$165.75	$208.25
First 15 Week Payroll	$4,717.50	$2,486.25	$3123.75
Finals Week Payroll	$250.75	$250.75	$250.75
Total Semester Payroll	$4,968.25	$2,737.00	$3,374.50

Difference between Examples:

$4,968.25 (Example 1) - $2,737.00 (Example 2) = $2,231.25

$4,968.25 (Example 1) - $3,374.50 (Example 2) = $1,593.75

RECOMMENDATIONS

I recommend that you consider the use of both unpaid and paid student lab help for the following reasons. First and foremost, the system provides an opportunity for us as lab managers to give more students an additional learning experience if they so choose to work in the lab. Secondly, the system helps all those involved: the students, the lab management, the lab workers and the lab instructors. Thirdly, the system better utilizes your student lab workers. Finally, the system allows you to keep your labs open even more hours for less money.

REFERENCES

1. Payne, Michael. "Effective Use of Student Employees In A Microcomputer Lab," Proceedings of the 1991 ASCUE Summer Conference, (June 1991), pp. 180-183.

2. Stallworth, Karen. "Quality Control in Student Lab Assistants," Academic Microcomputing Conference Proceedings, (April 1990), pp. 309-316.

3. Williams, F. and Whitkanack, G. "Talent, Trust, and Training: Keys to Managing with Students," Academic Microcomputing Conference Proceedings, (April 1990), pp. 309-316.

VOLUNTEER STUDENT AGREEMENT

FALL 1992 SEMESTER

I agree to work during the assigned hours as shown in the schedule unless there are extenuating circumstances. If extenuating circumstances occur, I agree to find a replacement and report my absence to the CPT Lab Manager.

I agree to abide by the rules of the CPT Labs.

I agree to follow the instructions of the Lab Manager in accordance to his/her duties as a manager and my duties as a student lab assistant.

I agree to perform the following duties:

Report any hardware/software problems associated with any of the CPT labs to the appropriate responsible person.

Keep the printers loaded and running in the CPT labs

Keep the CPT labs clean

Boot\reboot personal computers as needed in the Hofferlab.

Report any student that does not abide by the rules of the CPT Labs.

By signing this document, I am attesting to have read it and understood it and agree to abide by its statements and rules. I am also attesting that I am not on academic probation.

_____

TRAINING SESSIONS - All CPT lab consultants are required to complete 2 training sessions of 2-3 hours each. Please check your folder in the CPT office for the actual times and locations after you return for the Fall 1992 semester.

VOLAGREE.DOC

272

Multi-user Access In a Campus Network: The Next Generation

Mary L. Smalls, Assistant Dean
Library and Information Services

Stephen E. Glover, Director
Information Resource Management
7313 State College
Orangeburg, SC 29117

INTRODUCTION

Historical Mission of the Institution

South Carolina State University, a senior college, was originally established in 1872 in compliance with the First Morrill Act of 1862. It was named South Carolina Agricultural and Mechanical Institute and existed within Claflin University. Later, to comply with the Second Morrill Act of 1890, the South Carolina General Assembly enacted legislation for the severance of the South Carolina Agricultural and Mechanical Institute from Claflin University. As a result, the Colored Normal, Industrial, Agricultural and Mechanical College of South Carolina was established with a distinct land-grant mission. Then in 1954 the name of the College was officially changed to South Carolina State College by an Act of the General Assembly. On February 26, 1992, the name of the institution was changed to South Carolina State University (SCSU) to reflect the State of South Carolina's adoption of the concept of "the teaching university."

The University is located in Orangeburg, South Carolina which is in the central region of the State. It is situated in a tri-county area that is undergoing a rapid transition from a traditionally rural, agricultural economy to one that has a business, industrial focus which is national and international in scope. Thus, as this area of the State moves to enhance its economic development, SCSU is committed to increasing its participation in this vitally important process by continuing to expand its national and international programs.

As the 1890 land-grant institution in South Carolina, the University is a state-wide resource which has historically served the African-American citizenry of the State. It continues this tradition by offering educational programs which produce grad- uates who are prepared for effective participation in local, state, regional, and national locations.

With its traditional focus on teacher education, research, and service, SCSU's mission is to provide undergraduate in- struction which combines liberal arts with vocational education. Undergraduate students are provided a broad range of academic majors for which they can earn a baccalaureate degree in one of the following areas: engineering technology, agribusiness, home economics, human services, health sciences, education, business, and the arts and sciences.

Similarly, the University offers a select number of graduate degree programs. Degree programs at the master's level are predominately professional programs in teaching, human services, and agribusiness. The doctoral and specialist degree programs in educational administration emphasize the determination of the University to maintain a balanced expression into advanced study as well as to ensure adequate resources to support quality undergraduate programs.

As a part of its expanded mission, SCSU considers the provision of community services vitally important. These services are provided through 1) programs related to agriculture, 2) adult and continuing education, 3) research, and 4) reference and resource areas. Each of these services is designed to create a mutually beneficial relationship between the community and the University.

SCSU is fully committed to its mission which is to provide for all students quality academic programs and the research initiatives to support these programs. The ultimate objective is to ensure that all students are able to work and live pro- ductively and successfully in a rapidly evolving high-technology society as a result of their educational experiences at the University. The institution has an enrollment of over 5,000 students.

Early Developments of the Campus Network

Ten years ago, SCSU (then South Carolina State College) began its emergence from a 32 KB keypunch driven Computer Center. At best, computer services at SCSU serviced only a segment of administrative and academic users. During this period, the institution made a conscientious effort to meet the needs of its academic and administrative users by upgrading its computer resources.

The academic community (to include department computer labs,faculty advisement and program management, 1890 Research, the library, engineering, self-study and print services), required greater access and support. The lack of a high speed fiber-optic infrastructure impacted negatively on research and instructional opportunities.

The campus-wide network also required isolated academic labs, faculty, and staff to share resources and exchange information files and programs whether located on campus, in the State, or around the world. The smaller computers in the network needed access to the powerful capabilities of larger systems, while the larger computers needed to take advantage of smaller dedicated systems chosen for specific application environments.

Conceptual Framework

During 1990-91, the University's campus-wide network wasborn. This concept reflected the institution's need to exchange ideas -- to share not only lessons learned from the past but also steps being taken to meet current and emerging needs -- and, perhaps, to speculate on the future.

Today, the Computer Center, known as Information Resource Management (IRM) serves as a support base for the entire campus with efficient, effective, and economical management of informa-tion and information resources. These resources include Computer Services, Print Services, Records Management, and Telecommunications, all of which are administered by the Director of IRM. IRM serves as both a resource management function within the University and as a support base for the University's overall management perspective on improving organizational productivity and effectiveness. Thus, IRM's strategic objective is to maximize the quality, use, and value of information within the university's environment.

The implementation of the electronic office concept was also considered essential so that all areas of the University could have access to online wordprocessing, electronic mail, informa- tion management, calendar functions, scheduling, and conferencing functions. The campus-wide high speed fiber-optic network infor-mation concept offered flexibility by allowing for the substitu- tion and addition of new technologies of future phases while maintaining compatibility with previous phases. Crucial to this process was the enhancement of the University's data network in support of the integrated online library system (IOLS). With the infusion of records on both electronic and hardcopy media, it was extremely important that the University have the capabilities to control, manage, retain, and retrieve records in an efficient and cost effective manner.

However, as technological plans were formulated for the administrative and academic campus users, the library staff was also making plans to automate their facilty and emerge into the mainstream of technological innovations. As a result, this paper focuses on the campus-wide network and the library as a forerunner in the network's design.

OBJECTIVES

The Campus Network

During 1986, the University conducted an internal assessment on computing to include a conceptual framework of the primary issues impacting computing at the institution. A recent assess- ment determined that SCSU continues to face demands from many areas of the University which require access to current resources and to have new services become available (e.g., Internet). The recommendations included in the assessment report were based on 1) an assumption of where the University would like to be in five years; 2) meetings with academic and administrative departments on campus; 3) consultation with vendors; and 4) visits to other universities to review how they plan to implement their emerging technologies. The assessment highlighted four primary objec- tives:

1- to provide greater latitude for staff and faculty to assess administrative management tools and to teach from a broader platform;

2- to provide students the ability to gain experience in a state-of-the-art environment, which will more nearly simulate the industrial environment they will be joining after completing their education;

3- to provide students with more tools to explore answers, and

4- to provide second and third level problem accessi- bility to students which will reduce the costs of experimenting and provide high reliability factors.

The Integrated Online Library System (IOLS)

Moreover, during the time the 1986 University assessmentwas being undertaken, the library staff was seeking consultant services that would assist in identifying their needs for improved user programs and services in a technological environ- ment. In conjunction with the library's plans, the Director of IRM became involved. His first responsibility was to assist in selecting a library consultant. This was superseded by his assignment to a Task Force that developed the needs requirement report and the RFP for the library.

As plans for the IOLS were defined in 1987, objectives were identified to guide the implementation of the project. Theywere as follows:

1) to increase the accountability of resources to users, as well as fiscal accountability to the college;

2) to provide comprehensive, statistical infor- mation needed for evaluation, planning, and management reports;

3) to provide students, faculty, and staff an indepth source of information in their pursuit of academic, research, and administrative needs;

4) to protect the bibliographic database of library holdings;

5) to provide access to resources from many user stations in the library for campus and community use instead of one physical location, and

6) to make effective use of personnel.

FUNDING

To fund a campus-wide network project required a major expenditure to ensure that a complete and comprehensive high speed fiber-optic backbone network was developed. Without federal support, SCSU could only at best develop a fragmented approach in providing technological resources to its students, faculty, and administrators. Administrators and others involved in the assessment recognized that funding solutions for today's network must encompass and address all factors: 1) size, 2) multi-vendor make-up, 3) diverse applications, 4) dynamic nature, and 5) unpredictable use. It was evident to SCSU that information was indispensable to competitive strength. Like money, it was an invaluable asset to the institution. Just as it was important to ensure the liquidity of monetary assets, it was crucial to the success of SCSU to ensure the liquidity of information -- that is to enable information to flow easily and quickly through the institution. For SCSU to become competitive with peer institutions, it must be in a position both financially and technologically to seize market advantage by providing a pathway so that information could be controlled, managed, retained, and retrieved in an effective and cost effective manner.

Toward this end, based on the internal assessment, State budget, and short falls, the University instituted a strategy that utilized funds from several sources: 1) a bond referendum - large item instructional equipment; 2) a NSF grant - connection to a regional network; 3) Title III - upgrade infrastructure to include the library's online system; and 4) internal funding to upgrade academic computer labs and software resources. The University allocated $750,000 (internal funding) the first year of the project, with future year amounts to be determined based on State appropriations and University generated revenue.

Title III

Various sources contributed to funding for the IOLS. Initial funding was received from Title III, Section B, "Strengthening Historically Black Colleges and University Programs." This source provided funding for five years, 1987-92, to cover expenditures such as wiring inside the library, tele- communication lines to the Computer Center, CPU, software, staff, terminals, pc's, printers, barcodes, archival tape processing, authority tape processing, and other peripheral devices. To date, funding has been approved for another five year period, 1992-97. Phase II, the new period, will focus on implementation areas not completed in Phase I, as well as informational enhance- ments to the IOLS. Funding for Phase I totaled $1,367,282. The administration also supplemented the purchase of a CPU.

A proposal for Title III funds from the Library Services Construction Act (LSCA) generated $20,000. These funds were used to complete retrospective conversion of the library's collection.

The Commission on Higher Education (CHE) supported the purchase of equipment from a proposal submitted to them. This award totaled $127,769.

SCSU's requirement to interconnect many computer applica- tions, both administrative and academic, in a University campus environment also mandated special funding. There were several buildings with current and future requirements that needed access to the high speed campus-wide communications network which could be controlled at the physical layer. All buildings on campus had to be able to eventually connect to this system, so there was a need for future growth and easy expansion of this communications network.

A single break in the system could not cause network fail-ure, and a fault tolerant system was required to protect the information on the network from corruption. Additionally, a high speed network was required, that provided the same 10 million bit per second access for all buildings and a migration path to future high speed fiber-optic communications hardware. In order to convert from the then current "twisted pair" technology, the University submitted a Title III proposal, "Implementation of a Fiber-Optic Infrastructure in Support of Joint Use of Facilities and Strengthening of Fund Management," to support this transition.

Moreover, these funds ($214,000) will be used over a five-year period, 1992-1997, to implement a high speed fiber-optic infrastructure at SCSU. The University will also install appropriate protocol software to support interconnectivity between academic labs.

National Science Foundation Grant

This grant represented a use of federal funds to connect SCSU campus-wide network to the National Science Foundation Network (NSFNET). Funds ($33,000) were utilized to cover the overall cost of the connection to the NSFNET to include: 1) gateway equipment, 2) associated communication hardware and software installation, and 3) leasing of communication circuits to connect to the mid-level network; Southeastern UniversitiesResearch Association Network (SURnet). The University pro- vided all funding associated with the local network, support staff, space, air conditioning, and maintenance.

EQUIPMENT AND SOFTWARE

The Computer Center

The campus computer configuration is supported by a VAX cluster to include the following:

CPU - DEC
- 1 - 3800
- 1 - 4000
- 2 - 6310
- 1 - 8300
- 1 - 8350

VMS operating system
Memory -Internal main - 192 MB
External - 12 GB

The Library

Hardware

71 Terminals - DEC VT320 - 65
DEC VT420 - 6
56 available for library users

46 Printers - DEC LA-75 - 25
DEC LA-75P- 3
IBM proprinter X24E - 18 (staff)
25 available for library users

21 Personal computers - IBM - staff
PS/2 25 - 10
PS/2 30 5
PS/2 50Z 6

1 Desktop publishing system - IBM PS/2 55SX

1 Laser printer - IBM personal pageprinter II

5 Workstations - OCLC - M386/16 - 3
M300 2

10 Handheld lasers - Intermec

NOTE: 81 terminals and printers are available for users to access MIL-LINE. This equipment is located on all four floors.

Software Arts & Letters (graphics)
Wordperfect, 5.1
Pagemaker
dBASE III plus, 1.1
Lotus 1-2-3, 2.1
Kermit
ATLAS - IOLS software by Data Research Associates, Inc., St. Louis, MO

COMPONENTS OF THE CAMPUS-WIDE NETWORK

Campus Labs

Each academic building has at least one computer lab. Some of these labs are now able to access the campus-wide network and transfer files between various LANS on the network. The remain- ing labs without access to the VAX will be incorporated under the aforementioned Title III proposal. The department LANs are dedi- cated to departmental specific applications. This concept avoids duplication and produces expertise within a given application or series of applications. A description of the labs and their locations follow:

- The School of Engineering has several UNIX base workstations which are utilized in the main lab with a minimum of 24 devices and connectivity to the mainframe. Autocad software is standard.

- The School of Business has three LANS (MacIntosh, AT&T, IBM). These labs serve as a resource area, instruction and independent study, for instructors and students. The diversity allows for compatibility with other labs on campus that students have access to in preparing assignments, etc. They are opened from 7:30 am - 9:00 pm No connection to the VAX is available at this time.

- The Department of Mathematics and Computer Science has two labs that support the computer applications course for all non-computer science majors. One is equipped with hard disk drives and the other floppy drives. There is no connection to the VAX.

- A lab in the Computer Center is available twenty-four hours to support wordprocessing and other applications on the network. No instruction is provided.

- The Department of Communications has a Computer Assisted Writing Center that provides instructional resources to students with noted deficiencies and/or problems in compositional development. Tutors are available to assist students with their assignments.

- The Department of Natural Sciences has two MacIntosh and one IBM compatible lab. One Mac lab is used by students in completing assignments and software development. It is available from 8:00 am - 5:00 pm and 6:00 pm - 10:00 pm daily. The other is used by students in physics to supplment course requirements. The IBM compatible is designated for chemistry students. It is connected to the VAX.

- The Department of Nursing provides a client simulated lab with Apple 2E computers that students use for tutorial and skill development. It serves as both a lab and classroom. No connection to the VAX is available.

- The School of Freshman Studies uses MacIntosh computers for students to augment developmental studies objectives assigned by the faculty is this area. Its use is for students in the School. No VAX connection is available.

- The lab in 1890 Research is used by the staff to support the statistical analysis and data management of research projects. It is connected to the VAX.

- The School of Education uses IBM PS/2 computers for undergraduate and graduate students in teacher education. The students in the Department of Rehabilitation Counseling also have access to the lab which is not connected to the VAX.

The Miller F. Whittaker Library

The Miller F. Whittaker Library is centrally located on the campus. It is organized into four units: 1) the Library Manage- Unit is responsible for the general organization, development, and operational procedures of the library; 2) the Collection Development, Interpretation, and Utilization Unit assesses and selects a quality collection and provides intellectual access through bibliographic resources, references, and instructional activities to schools and departments; 3) the Collection Organization, Preparation, and Bibliographic Control Unit establishes and maintains the organization and bibliographic control of all library resources to meet the informational needs of the user; and 4) the Circulation Unit establishes and administers all procedures relative to the circulation of library resources. The staff of twenty-eight is comprised of eight librarians and twenty support staff.

Users have access to 264,867 volumes of books, 1,332 perio- dicals, 48 newspapers, 66,157 federal documents, 11,409 state documents, and 520,705 microforms. Emphasis is placed on collec- tion development, dissemination, interpretation, and utilization of library resources to faculty, staff, students, and community users. Three levels of bibliographic instruction and the library-faculty liaison program assist in providing mechanisms for enhancing the library's programs and services.

The library is a partial depository for U.S. government publications, a subdepository for South Carolina government publications, and a regional data center under the auspices of the South Carolina State Data Center. These materials supplement information requests for census, statistical, and ephemeral data.

The library is also a member of six information retrieval systems. These alliances give users access to more than 10,000 library collections in the United States for interlibrary loan, access to subject databases for current and retrospective information via computer searches, and access to the actual document via the computer document delivery system. The systems are: 1) SOLINET (Southeastern Library Network), 2) BRS (Bibliographic Retrieval Services), 3) DIALOG, 4) VUTEXT, 5) EPIC, and 6) ALANET.

Offline access to five electronic indexes in CD-ROM (compact disk read only memory) format is available for users. These include: 1) InfoTrac (general interest magazines and The New York Times), 2) Wilsonline (Education Index), 3) Disserta- tion Abstracts, 4) Newsbank (newspapers), and 5) Academic Abstracts. Computer printouts of bibliographical citations, and, in some cases, abstracts are provided to users.

Full text access is CD-ROM format include ProQuest 1) General Periodical OnDisc, 2) Social Sciences Index, and 3) Business Periodicals OnDisc. These systems generate full text articles for eighty percent (80%) of the journals in the library's collection.

MIL-LINE, the integrated online library system, allows users to identify the library's books, state, and federal documents online, as well as determine the availability of materials for use. In addition, users have online access to sixteen indexes in CD-ROM format in MIL-LINE. These include:

1- Agricola
2- Agris
3- Applied Science and Technology Index
4- Biological and Agricultural Index
5- Biotechnology Abstracts
6- Book Review Digest
7- CRIS/ICAR
8- Education Index
9- Environment Library
10- ERIC
11- Food Science and Technology Index
12- GPO Index
13- Humanities Index
14- Nursing and Allied Health
15- Readers' Guide Abstracts
16- Social Sciences Index

Users also have online access to FEDIX, an online database retrieval service of government information for colleges, universities, and other organizations.

HUMAN RESOURCES IN SUPPORT OF THE NETWORK

SCSU is dedicated to the matter of developing an effective, comprehensive fiber-optic network, and through this process it allows employees to benefit from the educational challenges inherent therein. To facilitate the process, the institution utilized its own personnel resources in accomplishing this task.

The Director of IRM and his department expended major por- tions of their time coordinating and implementing this activity. Faculty and other staff members were utilized as resource persons during various phases of the activities. These persons had com- puter and/or network experience, as well as knowledge and back- grounds in analyses. See Appendix B.

In the initial planning stages of the IOLS, a library con- sultant was hired to ascertain data for a needs assessment, re- quirements report, and RFP. The Coordinator of Collection Organ- ization (Technical Services) was given the responsibility of Project Manager of the online system. A programmer analyst was hired to serve as liaison between the library staff and Computer Center staff. Some of her responsibilities also included train- ing and monitoring the various subsystems of the IOLS. Additional staff was hired to complete the retrospective conversion project, barcode the library's collection, input copy and volume information into bibliographic records in the online catalog, etc.

Network support within IRM includes Computer Services, Print Services, Records Management, and Telecommunications. The present and future configuration of the network is located in Appendix A.

Computer Services

The role of Computer Services is to provide support in the processing and management of the University's computer resources. Its three principal functions are 1) Administrative and Academic Systems Development/Maintenance, 2) Computer Operations, and 3) Technical Support. The staff consists of a systems manager, four programmer/analysts, an operations manager, a data coordinator, and three computer operators.

Print Services

Print Services is responsible for the University's printing and duplicating needs. The department also provides 1) print consultants - to advise and assist departments in printing speci- fications; 2) forms control - to review the requirements of departments for various kinds of forms and assist the Office of Records Management in forms control, development, and revision; and 3) mail insertion - to provide automatic mail insertion for large mailings. The print services staff consists of a print product manager and four print equipment operators.

Print Services capabilities were upgraded with an AFGA Compugraphic Proset 9400 compatible with the MacIntosh system. Additional equipment includes the MacIntosh IICI and Se/30 CPU, a Proset imagesetter, a laser printer, a scanner, and a rapling 43 film processor. It allows network users to transmit documents directly to the Print Shop.

Records Management

Records Management involves the process of systematic and scientific control of recorded information that is required in the operation of SCSU. Such control is implemented over the creation, distribution, utilization, retention, storage, retrieval, protection, preservation, and final disposition of all types of records within the University. The staff consists of a records analyst and an administrative specialist.

Telecommunications

Telecommunications deals with the electronic process that permits the passing of information from one sender to one or more receivers, with the output being in a usable form (printed copy, fixed moving pictures, and optical signals). It also includes all services, products, media, and methodology used to deliver information ranging from a simple telephone to sophisticated fiber networks.

The telecommunications staff has responsibility for the development and management of the facilities used by the University for voice and data communications which include:

- to determine voice and data lines, network, and software characteristics applicable for installation;
- to determine terminal and control equipment;
- to develop and enforce standards for design and installation of communications based applications;
- to assist system designers, system programmers, users, and management in voice and data communications, planning, and problem solving, and
- to maintain voice and data lines, networks, and terminal configuration.

The staff consists of a telecommunications manager, an administrative assistant, two PBX attendants, and one field technician.

NETWORK SERVICES

The Management Support Systems

These systems include:

- Human Resource System (HRS) - payroll, personnel, and employment data;
- Student Information System (SIS) - student admission, records, financial aid, billing, receivable data, and on-course module;
- Financial Resource System (FRS) - financial accounting, accounts payable, purchasing, fixed assets, property control, and the student loan system;
- Library Information System (LIS) - online access to library holdings, and
- Alumni Development System (ADS) - alumni, development, and tracking.

Network Integration Services

- FOCUS - 4GL provides access to management data for reporting purposes to network users on a "need to know" basis. Application probability between the VAX MacIntosh and personal computers convert data into a useful format for all levels of users
- 386Ware - allows v.t. terminals on the network to run pc application software and access CD-ROM's on the library's network
- Academic lab connectivity

. TCP/IP - provides academic LAN connectivity to the VAX

. File transfer - provides reliable bi-directional binary, machinery, and ASCII file transfers with wild card options

. Virtual disk - makes host disk space accessible to the micro-user. This feature reduces local disk requirements, allows multiple microcomputers to share data and enables microcomputer data to be automatically integrated into the host back-up procedures

. Network printers - enables local files to be submitted to any printer on the network, reducing the need to have a printer at every workstation

- Office Automation Facility - provides document processsing and transfer, electronic mail, time management, communication, desk calculator, calendar functions, and file cabinet management

- Modem Pool - allows the campus network users access to outside computer facilities and services; thereby, reducing the need for individual modems. It also allows users to access the campus network at off campus locations

- Network Server - manages remote LAN servers and provides common applications to be shared across the network

SURAnet

SURAnet is a wide-area regional computer network that provides access to the National Science Foundation Network and the Internet community for organizations throughout the southeastern United States and Puerto Rico. It was created to facilitate communication and collaboration among educators, scientists, and researchers in academic, governmental, and commercial organizations. These organizations have access to a vast number of resources including the National Super Computing Center.

OBSERVATIONS AND FUTURE PROJECTIONS

The past, the present, and the future impact our lives in one way or another because of change. Assessment, evaluation, transition, and transformation are all elements of change -- elements that impact the way we perceive, project, and make decisions.

As SCSU reflects on the past, addresses the present, and plans for the future, its technological environment will be in the forefront of its decision making process. Funding will be one of the major concerns in this process. It underlies the degree to which change can be implemented.

The State's plan for video technology will impact the role SCSU will take in this endeavor. A study conducted by the Bud- get and Control Board's Division of Research and Statistical Services concluded that there are four functions most commonly being satisfied by television technology in the State. These four functions are 1) communicating information in a timely manner; 2) providing a medium that enhances educational and training opportunities for employees of the State and its citizens; 3) reducing State government travel costs; and 4) promoting agency services and the State in general. The projected growth of television technology, as indicated by the findings of this study, point to the necessity of having an orderly and coordinated development of this technology to assure that resources are most effectively utilized and that unnecessary duplication in the areas of services, personnel, and equipment are avoided. South Carolina Education Television (SCETV) given its Statewide capabilities and its mission, is in the best position to address and monitor the growth of television technology in the State.

Communication linkages will broaden the geographic availability of information to faculty and students in the classroom. The new classroom has the potential for teleconferencing to distant learning environments, and the predesigned CD-ROM lessons on workstations may reduce the presence of an instructor.

The library of the future will need to focus on selective acquisitions of print and media products and the revision of collection development policies relating to various formats of information delivery. The outcome is to network individuals to a vast array of information resources. The advocacy role for future librarians is information providers, service providers, and technology providers.

As technology continues to infiltrate the academic community, organizational structure will also be a concern. For example, what will be the organizational structure between the director of computer resources and the library dean or director?

Furthermore, other challenges to consider include: 1) es- tablishing a users' reference library for all computer equipment, software, and network services; 2) determining licensing, leasing, and copyright issues relating to the transformation of information technology; 3) changing behaviors about technological applications; 4) integrating data with voice and video; 5) expanding networking applications with local, state, regional, and national agencies; 6) managing and administering the campus technology infrastructure; 7) implementing policies and standards for users; and 8) demonstrating and qualifying the results of information technology.

SUMMARY

Technology has changed SCSU in ways never envisioned ten years ago. Trends indicate that these changes will continue at an accelerated pace and perhaps, in ways just as unpredictable as those of the past. If we are to prepare students for the society of tomorrow, and position SCSU to take advantage of future opportunities, we must begin today to plan and develop strategies which will embed technology within the curriculum, as well as in offices. We must also adopt a policy of strategic management in addition to building the technological infrastructure.

At any rate, a number of important questions will develop:

. Are our policies and standards of today adequate to address the development of truly integrated systems?

. How will technological events impact our support of instruction and research?

. How can we maintain the rapid pace of bringing new technology and applications onto the campus?

. What challenges are posed as more and more information technology responsibilities are distributed across organizational boundaries?

. How should information technology managers address the growing issue of accountability?

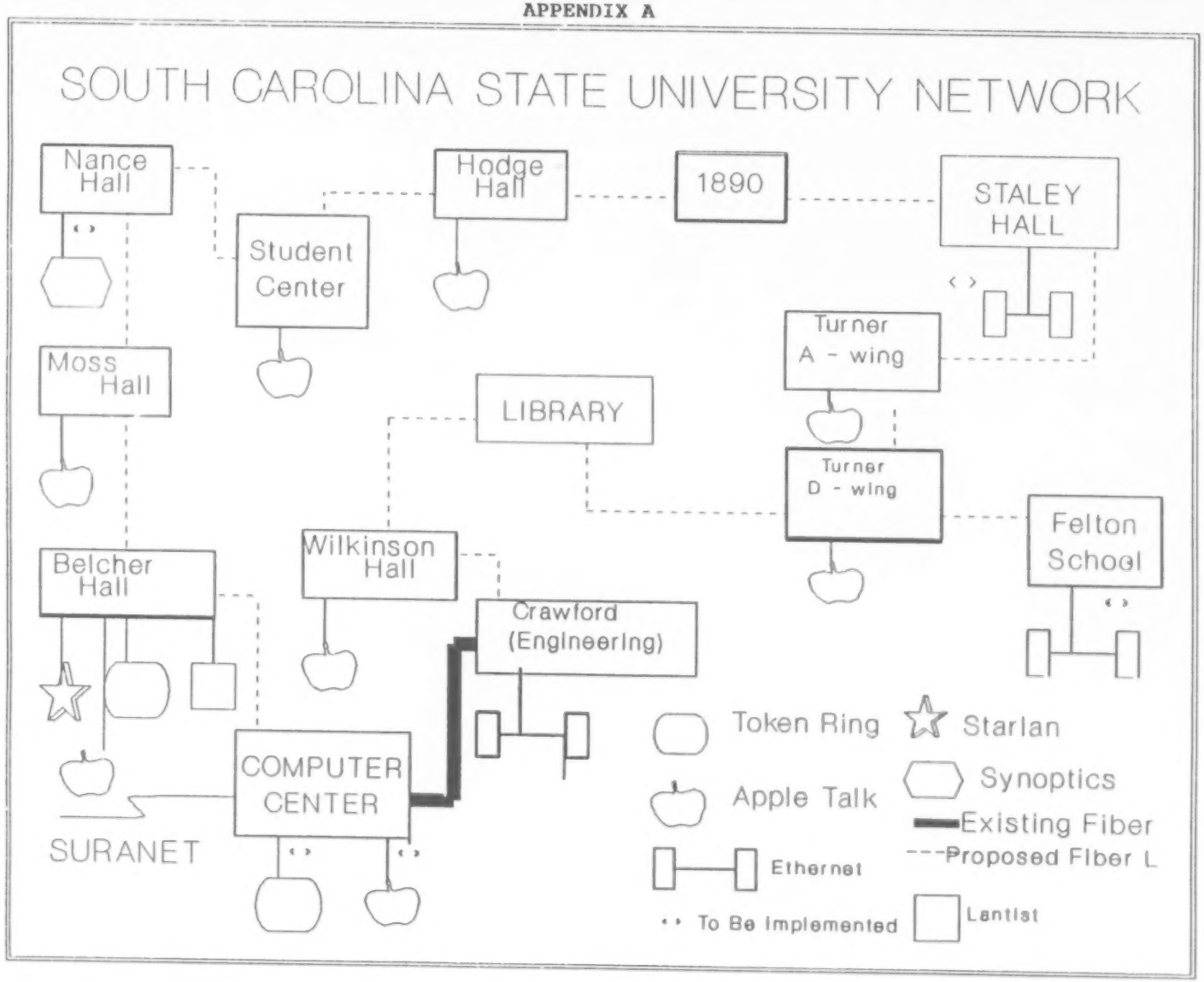
. What are the implications for managing information technology resources at SCSU?

. How will vendors structure strategic alliances with the University in the information age?

. How should we plan strategically as we look to the next century?

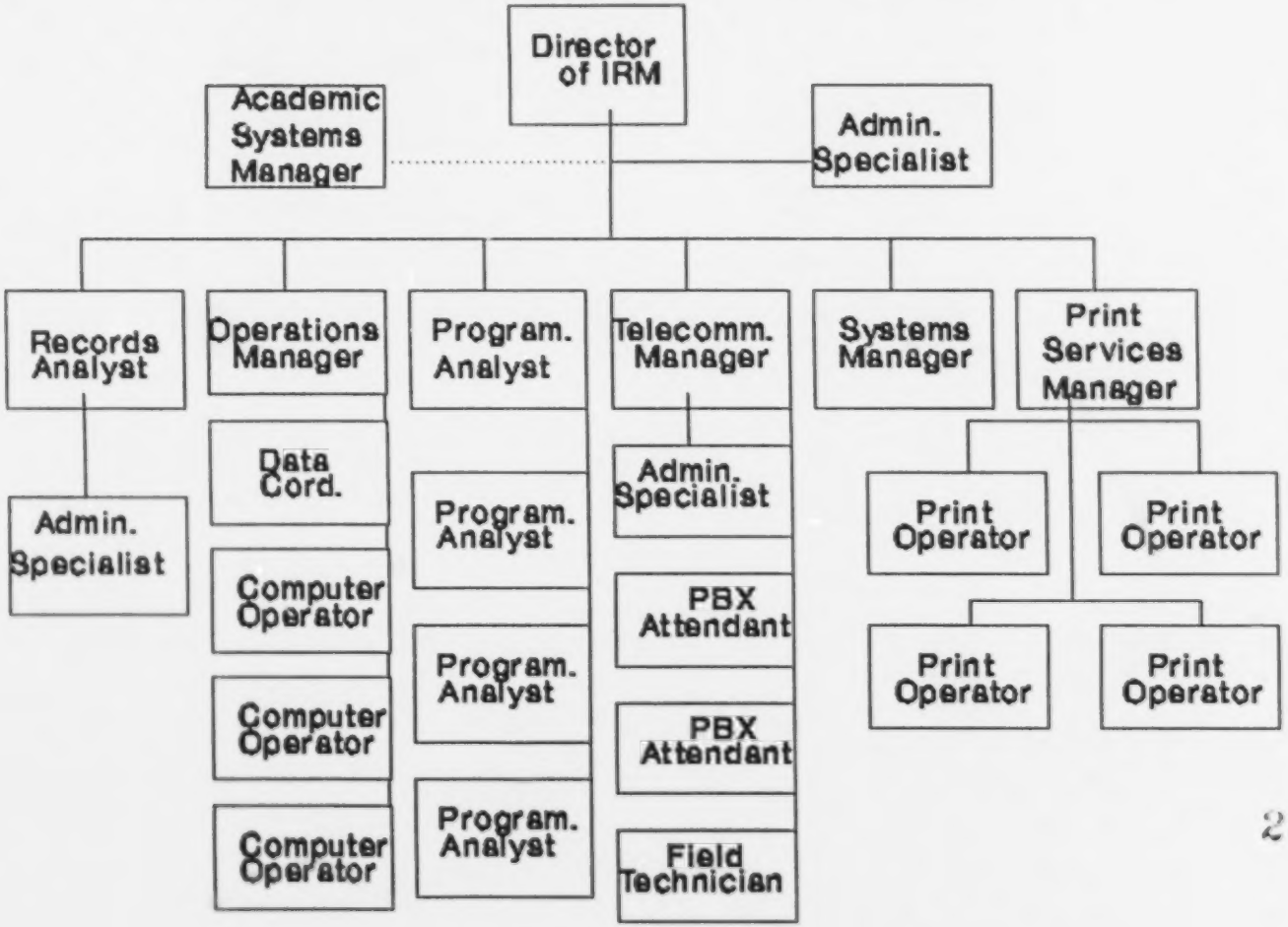
. How will organizational structure evolve as informa- technology becomes more persuasive on campus?

These are some of the questions that need to be addressed now and as future projections for information technology evolve at SCSU. How these questions will be answered centers around continued funding for existing technology, support for enhancements, and the acquisition of new products.


SOUTH CAROLINA STATE UNIVERSITY NETWORK
Nance Hall
Student Center
Hodge Hall
1890
STALEY HALL
Turner A - wing
Moss Hall
LIBRARY
Turner D - wing
Felton School
Wilkinson Hall
Belcher Hall
Crawford (Engineering)
COMPUTER CENTER
SURANET
Token Ring
Starlan
Apple Talk
Synoptics
Existing Fiber
Proposed Fiber L
Ethernet
To Be Implemented
Lantist

APPENDIX B

Information Resource Management



Utilizing Computer Labs Across the Technology and Other Curriculums

Richard L. Taylor
Purdue University North Central
Westville, Indiana 46391

ABSTRACT

The Purdue University North Central campus computer use has evolved and expanded these past few years with limited resources and space. Our computer use in 1986 was 16 courses with 413 students in 4 computer labs and has expanded to date to 61 courses with 1018 students in 6 labs. Some courses are taught in a computer lab while most courses are taught in a classroom situation utilizing a mobile computer with an overhead datashow system. Students are expected to complete computer assignments outside regular course time. Computer labs are managed on an open lab concept and actual in lab course time is minimized. A workable system has evolved that has allowed for maximum use of facilities with a minimum amount of student and faculty frustration.

CAMPUS HISTORY and BACKGROUND

Purdue University is the Indiana link in a nationwide chain of land grant colleges and universities. In addition to the main campus in West Lafayette, Purdue University has established a series of regional campuses for the purpose of extending educational opportunities to the major population areas of Indiana. Purdue University North Central, located on a 264-acre site just south of Michigan City in Westville serves communities in the north central part of the state.

Working within the general missions of Purdue University, the North Central campus is dedicated to providing a range of services to the residents of its service area. These services include the provision of educational programs leading to associate, baccalaureate, and master's degrees. The campus provides all the usual support programs including financial assistance and other student services. All 3550 North Central Students are commuters and many attend on a part-time basis. The average student age is about 29 years of age. The first one or two years of many Purdue University programs are available for students who wish to transfer to other campuses of Purdue University or to another university. A variety of courses, credit as well as non-credit, are offered in evenings and on the weekends to provide opportunities for the working student. The Bridge Program provides development courses to prepare working adults to enter or re-enter collegiate studies.

COMPUTER LABORATORY FACILITIES EVOLUTION

The academic physical facilities of the campus include a 90,000 sf Education Building (Schwarz Hall, 1967), a 100,000 sf Library-Student-Faculty Building (LSF Hall, 1975), and a 2,600 sf Robotics and Flexible Manufacturing Laboratory (FMSL Lab, 1985). A single computer laboratory with the data punch card equipment and a computer center mainframe served the campus through the 1970's.

The 1980's ushered in the personal computer age with the campus purchasing 20 TRS-80 Model III Radio Shack computers for the Technology programs. A second laboratory of 24 IBM-PC's was established in 1983 for computer courses and general campus use. As noted in the abstract the campus utilized 4 computer labs in 1986 serving 16 courses with 413 students and has since expanded to 6 labs serving 61 courses and 1018 students to date in the 1992 Fall semester.

The computer laboratory facilities as summarized in Table I include the six computer laboratories and typical uses as noted. Every faculty office on campus has a computer that is connected to the Sequent computer for "E" mail and the listed computational purposes. A second academic network is supported by an IBM PS/2 Model-80 file server with Novell software. Nine student and three faculty

computers are linked to this Novell network that is utilized only by Computer Technology (CPT) program courses.

MANAGEMENT PROCESS of COMPUTER LABORATORIES

The management and maintenance of all computer laboratories except the CAD lab is the responsibility of the campus computer center. The CAD lab is managed and maintained by the Technology/Engineering Section.

The unique process of managing the computer labs utilizes an open laboratory concept. Only English composition classes are taught in that computer lab on a regular day-to-day basis. The limited number of computer labs vs the large number of courses that want to utilize the labs for course oriented application software prohibit the use of any lab on a "day-to-day" basis to teach any course.

The Purdue University North Central computer lab use policy has evolved through the 1980's that all courses utilizing computer software will be taught in a standard classroom. The first written policy for the campus was established in 1987 to formulate a gentleman's agreement that had evolved up to that point in time. The 1987 reservation policy is attached at the end of this paper.

Two computers on wheeled carts and overhead "data shows", owned by the Technology/Engineering Section, are used by faculty in the classroom to demonstrate software, explain course assignments, and answer computer related questions. Faculty could supplement instruction by reserving a computer lab 2 or 3 times a semester to specifically instruct students in a "hands-on" atmosphere when necessary. Students are otherwise required to accomplish and complete computer related assignments on their own time in an open, non-reserved lab setting. Faculty reserve a computer lab from the computer center and said reservations are posted at the lab door.

The English composition lab is located in the library and is an open lab when not being used for a composition class. The CAD lab is reserved on a regular basis for the laboratory portion of CAD/drafting courses and other Technology courses. Both lab schedules are developed by each section when the semester schedules are established.

The computer center staff manage and maintain 5 of the six labs. Three staff members handle the bulk of the duties with a software technician, a hardware technician, and a work study student helper. DOS disks are available for student checkout from the computer center mainframe office. The CAD lab is managed by the Technology Section and maintained by the Technology Section lab technician.

The open lab concept has worked well for our campus during the 1980's. The Technology faculty have utilized the two portable computer data-show systems. Other faculty are increasingly requesting use of the systems. Recent experience during the Spring 1992 semester project some difficulties ahead. More faculty from other sections are seeking to utilize computer applications and assignments in their various courses. Some faculty who have not used the portable computer feel that they can only teach computer applications in a computer lab are pushing for adjustments. This increased demand for use of the computer center facilities by new faculty computer interests has resulted in a revision of the computer center microcomputer lab reservation policy. The 1992 reservation policy is Attachment II to this paper. Essentially, faculty may now reserve a computer lab for up to 8 class sessions per semester.

Evolution is an appropriate term as more faculty and students seek to utilize the limited number of available computer labs. Compromise and accommodation will be necessary for all involved to maintain a workable computer laboratory facility with limited resources.

OTHER PROBLEMS and FUTURE IMPROVEMENTS

A recent problem has been the management of hard disks in the two labs utilizing hard disks, the Training lab and the CAD lab. Virus contamination, game playing and erasing files has been a constant irritation with the introduction of these two hard disk computer labs. The Training lab uses a concept of filling the hard disk to avoid students loading their own software on the Purdue University North Central hard disk. The CAD lab can not use this technique since AutoCAD needs open disk space on the hard disk to process its created files. Both labs are checked routinely, every week or two, for virus contamination and improper files are removed.

The advanced age of the computers in some of our labs present hardware maintenance difficulties. The double disk drive IBM PC's, XT's and Zenith computers are at the end of their useful life. Equipment hardware failure is a continuous and constant frustration to students, faculty and staff.

Future considerations include new computers in several labs. A faculty and staff committee is considering what equipment to purchase that will accommodate the expanding uses of the next several years. Networking is being given serious consideration to eliminate the hard disk maintenance difficulties.

A new building is proposed for the campus that will include various technology labs and a new computer center complex. The new computer complex will include a 2400 sf open lab, two teaching labs, and three CAD labs for the Technology Section. Computer labs will be maintained in both other buildings on campus with at least two computer labs per building. This proposed new building is a hostage of the current economic crisis as the state legislative budget committee is debating approval of new state construction projects vs the economic health of the State of Indiana.

COMPUTING CENTER

MEMORANDUM

TO: Distribution

FROM: Dale Stewart

DATE: February 24, 1992

SUBJECT: MICROCOMPUTER LAB POLICY

Circumstances have made it necessary to revise the lab reservation policy. New software is available, and faculty interest has grown! We are pleased about this, but we do not have as much (nor as powerful) equipment as we wish. We are now involved in the development of a plan to upgrade all the labs. As we reviewed lab usage, we discovered several interesting facts:

For all labs, reserved time is 25% of available time.

Lab 25C is reserved 10 times as much as all other labs combined.

Training and Continuing Ed reserve 25C 6 times as much as all other users combined.

When the English department stopped using the LSF lab, it has received little use.

Clearly, there is a movement from the labs with the older machines to 25C, where more powerful machines are available. Lab 25C was purchased expressly for Staff Training and Continuing Education. We have permitted limited use by others, but the demands are great. We must maintain fairness to all users. Accordingly, the policy has been revised and a copy is on the next page.

We thank you all for your cooperation in the past, and trust you will continue to help us maximize the availability of our limited resources.

LAB RESERVATION POLICY Revised: 2-24-92

If not reserved by an instructor for specific class use, Labs are "open" for use by anyone. An instructor may submit a request for a lab reservation to the Computing Center secretary. You may do this using forms available from the rack by the computer room window. If approved, you will be notified and your reservation will be posted by the door of the reserved lab. During your reserved use of the lab, you have the right to ask others to leave, or you may welcome others to quietly remain or enter if there are available workstations. You may bring your class into a lab without a reservation, but you may not ask others to leave, and you may not disturb those who are working there. Individual lab policies are listed below.
LSF Lab - May be reserved by any class for a maximum of 8 sessions per semester (first come, first served).

SWRZ 25A - May be reserved by any class for a maximum of 3 sessions per semester (first come, first served).

SWRZ 25B - Computer literacy classes have first priority and may reserve for a maximum of 8 sessions per semester. Other classes may reserve for a maximum of 3 sessions per semester (first come, first served).

SWRZ 25C - Priority is given first to staff training, then continuing education. These may reserve for all of their class sessions. Subsequent priority is given to academic classes, which may reserve the lab for a maximum of 3 sessions per semester (first come, first served).

SWRZ 35 - May be reserved only by CPT classes which use the devices located there for a maximum of 8 sessions per semester (first come, first served).

COMPUTING CENTER

****
MEMORANDUM
****

TO: Distribution

FROM: Dale Stewart

DATE: October 2, 1987

SUBJECT: **MICROCOMPUTER LAB RESERVATIONS**

Since last year, several changes have been made to the microcomputer lab facilities. Attached are revised sheets describing our facilities and the lab reservation policies and procedures. As we continue to expand our services and facilities, it is our hope that you will find us better able to serve your needs.

Our lab reservation business is becoming fairly large, and we must remind you that all reservations must be requested in writing to avoid confusion, misunderstanding, and delays.

Please read carefully the attachments and file them for future references. If you have any need for clarification or information in other areas, please do not hesitate to call me.

MICROCOMPUTER LABS PAGE 1

LOCATION	QUANT	TYPE	MEMORY	DRIVES
LSF 243	24	ZENITHPC	640K	DOUBLE
SWRZ 25A	10	IBM PC	256K	DOUBLE
	7	IBM PC	256K	SINGLE
SWRZ 25B	16	IBM PC	256K	DOUBLE
SWRZ 25C	16	IBM PC	512K	DOUBLE GRAPHICS DISPLAY
SWRZ 25D	8	APPLE IIc	256k	DOUBLE
	8	TANDY TRS80	32K-48K	DOUBLE

AVAILABILITY

Labs will be available as "open" labs whenever the building is open. For the SWRZ Labs, software may be checked out through the Computer Room window from 8:00 a.m. to 10:00 p.m. Monday through Thursday; 8:00 a.m. to 5:00 p.m. Friday; and 8:00 a.m. to 3:00 p.m. Saturday. For the LSF Lab, software may be checked out at the library desk from 8:00 a.m. to 9:00 p.m. Monday through Thursday; 8:00 a.m. to 5:00 p.m. Friday; and 9:30 a.m. to 2:00 p.m. Saturday.

LAB RESERVATIONS

Under certain conditions, a lab may be reserved for a special session. During this time, the lab will be "closed" to general use. Because this can be a source of frustration to a student who needs to use the lab, policies have been developed to regulate lab reservations.

POLICIES

1. The Computing Center is responsible for all lab reservations and will process requests, arbitrate conflicts, confirm reservations, and post a weekly schedule for all computer labs.

2. Although each lab will have a posted reservation schedule, students frequently fail to see it. The person who requested the reservation is responsible to (gently) ask any users to leave at the proper time. Optionally, users may be permitted to continue working in the lab if the class for which it is reserved does not require every machine.

3. No unscheduled "reservations" or evictions are permitted.

4. An attempt will be made to avoid concurrent reservation of more than one lab, and in no case shall all IBM PC labs be reserved simultaneously.

5. Labs are not to be reserved in blocks or in anticipation of a possible need. Accordingly, reservations for more than one or two single isolated class sessions must be approved by the requesting department head.

6. Reservation requests are to be directed in writing to Sheri Koster in the Computing Center. They must be submitted no later than 1:00 p.m. Friday of the week before the requested date. Appropriate forms are available in the Computing Center or at the Support Services counter (see attached samples and forms).

7. No reservations will be made for a given semester more than one month before classes start, except in the case of department requests to schedule a class in the laboratory for a classroom.

8. All lab reservations will be made in accordance with the following established priorities.

PRIORITY	ROOM LSF 243	ROOM SWRZ 25C	OTHER LABS
1	English Department	Staff and Faculty Development	All Departments
2	All Other Departments	Continuing Education	
3		Rental to Industries	
4		All Other Departments	

MICROCOMPUTER LAB RESERVATION

NAME: _____ PHONE: _____

DEPT-NUMB-DIV: _____

CIRCLE DESIRED LAB: SWRZ-25A SWRZ-25B SWRZ-25C SWRZ-25D LSF-243

PLEASE BLOCK IN REQUESTED TIME(S) AND INDICATE DATE(S)

	MON	TUE	WED	THU	FRI	SAT
7:00 AM						
8:00						
9:00						
10:00						
11:00						
12:00 NOON						
1:00 PM						
2:00						
3:00						
4:00						
5:30						
6:30						
7:30						
8:30						
9:30						

COMMENTS: _____

TABLE I
Computer Laboratory Facilities
Purdue University North Central

LAB	HARDWARE	QUANTITY	SOFTWARE USED	AUDIENCE SERVED
CAD	ZEOS 386SX 8 MB RAM 40 MB HD IBM PS/2 Model 50Z 3 MB RAM 30 MB HD	9 8	AutoCAD V 11.0 Softdesk Arch. CADD Softdesk Civil CADD Timberline SLAM-SIM P-SPICE Primavera	All CAD/Drafting Courses Construction Courses EET Courses IET Courses
ENGL. COMP.	IBM-PC (Overhead Projection System)	22 1	WordPerfect (Student Version)	Freshman Composition Students
Training/ Academic	ZEOS 386 SX 2 MB RAM 80 MB HD (Overhead Data Screen)	16 1	Turbo "C" Hypertex Library Windows, Micro Cap PC:Solve Database 4 P-SPICE Quattro Lotus WordPerfect MS Word Paradox Human	1. Staff Training 2. Continuing Education 3. Academic Students a) Programming b) Math/Physics c) Business Law d) EET Courses e) Biology
Computer Literacy	Zenith 640 K RAM (2) 5 1/4" Disk Drive	24	MS Works (Current) Have used Framework, WordPerfect and Lotus (All student version software)	Literacy Courses
CPT Programs	Terminals attached to an IBM AS 400 IBM PS/2 Model 50 networked	20 5	COBOL RPG Framework Turbo "C" Turbo Pascal	Computer Technology Program
General Purpose	Terminals attached to Sequent - 32 MB IBM PC/XT IBM XT networked	12 12 4	SPICE "C" Fortran E-Mail Pascal	EET Students, Faculty Faculty and Clerical Offices

Empowering Student Employees in a Down-Side Economy

Stephen A. Vieira
Clark University
Worcester, MA 01610

Abstract

In these fiscally-difficult times, every institution must devise creative strategies for fulfilling the promise of support and service delivery without additional full-time hiring. Clark University, through its Information Resource Center (IRC), has mobilized its student employees in the last two years to provide service to the campus in a number of ways through an extensive and empowering system. Students at Clark perform a variety of duties from peer hiring, payroll, and evaluation to computer hot-line and technical support to computing facility management. The presentation will discuss and offer alternatives for providing training including external vendors, software written in support of this effort including payroll and tracking programs, and student contracts, handouts, and booklets.

Elements for a Successful Beginning

In the spring of 1990, the Information Resource Center (IRC) employed about 25 students serving as part-time consultants in the public computing facilities. Two FTE staff members moved to other positions leaving a significant gap inthe support of academic computing on campus. Three volunteers from the student consultant staff saw an opportunity to assist in the operation and to take an innovative step toward having some hand in the direction of the department until new FTE could be hired. Among their immediate goals and functions were the development of new resources within the computing facilities and the training and stabilization of student staff members working there.

In September of 1990, I was hired as the Manager of the Information Resource Center (IRC) and given the task of creatively restructuring the support of academic computing on campus, including the public computing facilities. Building upon the development of these three volunteers, and with the support of the Director of Information Systems, we devised a strategy to further delineate their position description and functions, to design a comprehensive program of empowering students, to offer new services never before envisioned being done by students, and to provide students an opportunity to learn and develop skills not necessarily available through their classroom activities. We call this empowering the student employee and we believe it is a unique set of experiences that are most valuable to and appreciated by students.

Students Employed in a Variety of Ways

As previously mentioned, the IRC employs about 25 students as consultants in the computing clusters on campus. Among their duties are hardware and software support, documentation, and training. I am certain that every institution uses students in a similar role. What I think distinguishes our university is the software programs we have written in support of these functions and the additional functions we request of students.

Each student is given an additional consultant account designed to give them additional diskspace in which to work but also as a tracking system for us using e-mail and a home-grown time clock system. When they come to work, they login to this account and an on-line data file tracks time in and out. The IRC accounts are also used extensively for technical updates, e-mail exchanges, schedule swapping, and problem tracking. Finally the account setup guarantees maximum communication for those people who work in the wee hours of the morning only. It would be impossible to track and advise these people in any other way without this system.

Each consultant is required to sign a contract very professionally designed which spells out carefully what the duties are, the expected responses by employees, and the support that management can provide. Along with

this contract, a handbook is provided that describes in detail some of the common practices and procedures which constantly come into question. Each student is required to read and sign the last page of this booklet delineating their understanding of what they have read. We reserve the right to ask for assistance with additional duties that might arise during the year.

Some of the additional duties we have asked this group to fulfill with significant training provided both in-house and from the outside have been memory upgrades for older computers, ethernet board installations, specialized cable creation, and network setups. We occasionally use the student consultants as a test base for new software and hardware setups as well. In this role, we get valuable feedback without impacting the entire student population and we get real-time information rather than off-hours statistics which sometimes do not accurately account for peak usage occasions.

From the body of this consultant group come three very special people every year. The positions are two supervisors and a workshop coordinator. Their responsibilities are reflective of the original three volunteers and their ideals for the roles that they played.

The supervisor role is continuing to dynamically change as the computing environments evolve. From interviewing and hiring new employees to reserving the facilities for faculty use for their classes to debugging network problems within the cluster, these people are the primary human resources and technicians for the Information Resource Center (IRC) in these public settings. They simply maintain the areas and make sure everything is running and that people using them can do what they need to do. It is the supervisor's role, in conjunction with the Manager of the IRC, to guarantee that people are employed who can perpetuate the program and offer even more flexibility for the future. The supervisor controls schedules, payroll, and evaluations. One of the best software developments presently in operation is a well-devised system for automating the process of payroll to take a six hour per week job and make it about one-and-one-half hours per week. The supervisor monitors the pulse of the computing clusters and must be able to be strong enough to know when changes are in order. The Manager of the IRC plays a significant role in viewing this position as nearly equitable to an FTE in that the information from this person is extremely valuable and must be evaluated in that fashion. The supervisor then becomes an overseer of the computing facilities and thus the key ingredient in change, in design, and in implementation for today and the future. Luckily the four individuals playing this part so far have been excellent and that has led to the selection process between the present supervisors and the Manager of the IRC for new supervisors as very demanding upon the candidates. We have several applications that are reviewed and extensive interviews are held in order to pick the right person. It is certainly a much desired role within the IRC as it really holds interesting alternatives for the future.

The workshop coordinator has the primary responsibility for setting up and holding workshops designed to familiarize students with the various software packages we have and how to access them. Additional workshops addressed during the school year have to do with resume design and construction, cover letter creation, and integration of existing software packages into a working set of tools. The workshop coordinator schedules consultants and facilities working closely with faculty interested in having their students get instruction on using a variety of packages. This role is extremely important in that first year students get a comprehensive account of what is available and how to use each. Additionally, as new software is introduced, development of training materials becomes another responsibility that must be fulfilled. All handouts and documentation provided through the computing facilities is screened by this position and approved for circulation. The use of the computing clusters in an educated fashion then becomes the primary goal of this person and the evaluation instrument for selection of the incoming workshop coordinator. The workshop coordinator has become a very valuable resource for faculty and as a result some irreplaceable feedback has been developed from this position. The three members of this team play a significant role in the year-to-year alterations and developments made in the computing clusters.

The successes of this group have led to the inception of two new programs which are about one year old. The first, the student technician position, has given us an increased flexibility in our on-site technical support

we can offer the user community. One of the first positions that was eliminated during the initial budget review was a part-time on-site technician for computers. Alternatives for providing this service were extremely limited and would have to come from existing staff members. In recognition of the need and the desire to be able to provide some responsive program for our clients, the IRC negotiated a comprehensive outsourcing computer maintenance plan with Digital Equipment Corporation which has been described by representatives of the company as a first. A significant cost savings was realized by the university in return for on-site trouble-shooting provided by members of the Clark community. DEC agreed to train several individuals on how best to provide front-line support and the IRC agreed to provide that front-line mechanism in order to minimize unnecessary technician visits.

The perfect match for this front-line group was a gathering of students who had experience working with networks, clones, Macintoshes, and user groups who understood the inner workings of a computer. It was not too difficult to find a few of these people and we sent them off to DEC for the appropriate training. The people we have had have taken the program to new heights in that we have actually been able to provide some on-site training of our own, devised strategies for debugging problems that have occurred in multiple instances, and created a database of problems as a tracking system of where problems have occurred, who has had the problems and the solutions we have provided to those clients who have needed them. Independently we have also provided memory upgrades, board and modem installations, and computer setup as other services related to these technicians. Two other functions we are looking into at this time will be inventory and preventive maintenance calls for this position if deemed a useful facet of the job.

Hot-line support has always been a difficult task for the IRC due to the sheer number of calls that come in per day and the time it takes to address each of the problems with some kind of reasonable response. In looking at the amount of phone time incurred, the missed messages with no voice mail alternative, and the increasing collection of in person problem solving being done, it was evident that we needed to implement some kind of mechanism that would best serve both our community and the IRC in a more efficient manner. It seemed to be a perfect match for students with some phone answering skills and perhaps some ability to be trained with some technical information. Given this proposal, we decided to institute a hot-line support mechanism.

Students were selected who could use the phone effectively to get information valuable to a person trying to solve a problem. These students answer the telephone, provide front-line support where they can, redirect calls where a technical problem is too difficult for them, and log all calls that come into the IRC. This on-line information is used to analyze the types of problems evident on campus, who is having them, and what proactive training and information could be used to alleviate their numbers in the future. The hot-line group also provides a much nicer voice mail alternative in that if a staff member is not available, a student takes the message and transfers it through e-mail to that staff member. We have had much positive feedback from people using this service and staff members who would not normally have received the message in the past.

These are the ways in which we are using students in a responsible manner at Clark University at this time. You may ask how we keep these programs going and I feel that initially we are lucky to have such devoted and interested students. I also feel we have a pretty good program for motivating students to participate.

Keeping a "Good Thing" Going

Motivation is such an intangible subject in that it is measured by the individual and not for the crowd. We have found that different students are encouraged to take part in these plans for any number of reasons. It is our belief that a great part of the motivation comes from a self-driven image of being part of something very exciting and dynamic. Here are other motivational tidbits we have for our interested students.

One of the natural rewards for performing these functions is, of course, fiscal. We try to pay our students an above average wage from other campus jobs. The job itself holds regular hours and the student can plan

appropriately for what they wish to make per week without exception. Another nice facet is that we do not expect student employees to work on holidays and special campus days. Full-time staff pick up the work to be done during those times.

A second payment of sorts is the additional quota attached to a consultants account. Very few others on campus have more than one account and students often view this benefit as extremely valuable when completing projects or saving e-mail messages. Though this might not seem to be a major reward, to some of these students, this is a motivational factor.

Another reward provided to all student employees, and one they value because it is something they cannot get anywhere else on campus, is a series of inservice workshops provided by the department on how best to utilize every software package in an integrated approach and how to use every piece of hardware in any computing facility. If it has to do with computing in the computing clusters at Clark, the inservice workshops provide this training.

Some of the incentive plans we have utilized include field trips to other sites where something unique resides that our students would not necessarily get a chance to see on campus, twenty-four hour access to the computing clusters at Clark, vendor demonstrations of new software or hardware that they would not normally be scheduled to see as a regular student, and field trips to conferences, shows, training sessions, etc., that are usually reserved for full-time employees. In this way we indicate to students that they are contributing members of our department and that their talents are viewed as being a major contribution to our efforts.

There are several intangibles in measuring motivation for student employees. Assistance is given to our student staff in creation of resumes that get interviews. We provide powerful learning experiences that cannot be duplicated in a simulated setting. Several internship program contacts have been developed over the past two years with outside vendors who have heard of what we are doing and students find doors open where roadblocks once existed. After graduation, both recommendations and employment agency connections get students further along the hiring process and often lead to fine positions with other well-known companies.

We do not profess to have all the answers to how to motivate students as it is a very individualistic and inexact science. It is our belief however that if a person has some interest we can find something that will further involve the expertise and technical know-how of the individual.

The Future and What It Holds

In closing I would like to say that we have been looking into further developing our program to bring in new students to be computer operators, print shop personnel, newsletter developers/authors, computer pre- and post- sales representatives, presentation setup teams, and statistical support groups. Our goals remain the same; to give students an opportunity to do something interesting while providing a valuable resource to the user community. Our future incentive programs might also envision more functionality and privileges for students while also looking into other on- and off-campus opportunities for them to get valuable learning experiences. I think we are empowering our students with unique educational and developmental opportunities and in these fiscal times they give us a set of tools for which we are extremely grateful.

Connecting to the Internet: A Look at the Services and the Tools to Use Them

William P. Wilson, Ph.D.
Advanced Software Support Coordinator
Gettysburg College
Gettysburg, PA 17325
Wilson@Gburg

What is the Internet?

The Internet is a large collection of networks that are tied together so that users of any of the networks can use the network services provided by TCP/IP (a computer communications protocol) to reach users on any of the other networks. The Internet started with ARPANET, but now includes such networks as NSFNET, NYSERnet, PREPnet, and thousands of others. There are other major wide area networks, such as BITNET, that are not based on TCP/IP protocols and are thus not part of the Internet. However it is possible to communicate between them and the Internet via electronic mail because of mail gateways that act as "translators" between the different network protocols involved.

A note on connectivity is in order. Your institution would pay for a high-speed data line to your nearest network Internet hub; Gettysburg is connected via a 56Kb line to Harrisburg, for example. These lines are leased from the phone comapny for some flat yearly fee, so regardless if one person or thousands of people make use of the connection the fee would be fixed and the same. These speeds provide for peppy interactions with the network as we'll see a little later.

So, the Internet allows new or expanded access to the following types of resources:

a) mailing lists
b) library catalogs
c) special purpose databases
d) file transfer from archive sites

Becoming familiar with the Internet is not so much a question of learning new skills since the skills necessary are really quite minimal. The much more difficult problem with the Internet is learning where things are that are of interest to you. Our purpose in this workshop is to expose you to what general types of information are available and the tools which are available to help you learn about where to find those sources of information that are germane to your work and interests. All the tools discussed here are free; they are public domain programs, which for many of us is a critical factor.

The Tools

Software	Machine	Function
Lists/Mail		
Discussion List	All	Archive of Interest Lists to choose from
Eudora	Mac	pop mailer for the Mac
POPmail	DOS	pop mailer for the DOS world
NUpop	DOS	another pop mailer for DOS world
Libraries/databases		
Telnet	Both	Terminal connection to libraries, other databases
Hytelnet	DOS, Unix, VMS	Telnet Accessible site information

Gopher	All	Gopher protocol
File Transfer		
Archie	All	Locate files of Interest
Fetch	Mac	File transfer for the Mac
ftp	DOS, Unix	File transfer for other platforms